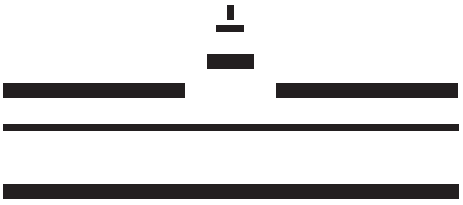

Westfälische
Wilhelms-Universität
Münster

Analyse sicherheitskritischer Aspekte einer Kläranlage mit Methoden des quantitativen Model Checkings

Masterarbeit

vorgelegt von:

David Könning

Matrikelnummer: 380977

Studiengang: M. Sc. Informatik

Thema gestellt von:

Prof. Dr. Anne Remke

Münster, 30. November 2016

Inhaltsverzeichnis

1 Einleitung

Die Versorgung von Menschen mit Strom, Gas und Wasser stellt heutzutage eine essenzielle Grundlage in westlichen Ländern dar. Um eine konstante und problemlose Versorgung zu gewährleisten, müssen diverse Faktoren berücksichtigt werden. Besonders die Versorgung mit sauberem Trinkwasser ist für das Überleben essenziell. Um eine solche Versorgung zu gewährleisten zu können muss zu stets eine ausreichende Menge an unverschmutztem Süßwasser in Form von Seen, Flüssen oder Grundwasser zur Verfügung stehen, welches dann zur Verwendung als Trinkwasser aufbereitet werden kann. Um dauerhaft die Reinheit unserer Süßwasservorräte zu sichern, muss dieser vor Verunreinigungen bewahrt werden. Aus diesem Grund werden heutzutage hochmoderne Kläranlagen zur Reinigung von Schmutzwasser eingesetzt bevor dieses zurück in die Natur geleitet wird.

Die Tatsache, dass bereits eine geringe Verschmutzung des Grundwassers gravierende Folgen für die Bevölkerung haben kann, macht die Zuverlässigkeit von Kläranlagen zu einem wichtigen Thema in unserer heutigen Gesellschaft. Um die Zuverlässigkeit von Kläranlagen zu gewährleisten, können diverse Methoden verwendet werden. Ein in der Wirtschaft sehr verbreitetes Verfahren ist die sogenannte Failure Mode Effect Analysis kurz FMEA bei der einzelnen Komponenten verschiedene Werte zugewiesen werden, die Ausschluss über die Häufigkeit, die Detektionsrate und die Schwere von auftretenden Fehlern Aufschluss gibt. In der Vergangenheit wurde die FMEA Analyse trotz ihrer großen Präsenz in der Wirtschaft immer wieder aufgrund ihrer wagen und ungenauen Aussagekraft kritisiert. Eine weiteres Verfahren, welches durch die stetige Steigerung von verfügbarer Rechenleistung immer mehr an Popularität gewinnt, ist das sogenannte quantitative Model Checking. Beim quantitativen Model Checking werden realweltliche Gegebenheiten oder Komponenten in Form von Modellen abgebildet. Die Auswahl von möglichen Modelltypen ist hierbei sehr groß, jedoch sind besonders verschiedene Formen von Markov Ketten bei der Modellierung durch ihre stochastischen Eigenschaften sehr beliebt. Über die so erstellten Modelle können dann

verschiedenste Aussagen anhand von formalen Logiken wie die Continuous Stochastic Logic (CSL) getroffen werden. Im Falle des quantitativen Model Checkings geben diese Aussagen nicht nur Wahrheitswerte zurück, sondern auch explizite Wahrscheinlichkeiten zu gestellten Problemen wie der Ausfallwahrscheinlichkeit eines bestimmten Moduls.

Diese Arbeit soll nun ein Übergang von der bereits in der Wirtschaft sehr etablierten FMEA Analyse zu einer Analyse basierend auf Methoden des quantitativen Model Checkings herstellen. Die Arbeit ist wie folge gegliedert. Zunächst wird in Kapitel 2 der Aufbau einer Kläranlage als hier verwendete Anwendung erläutert. Kapitel 3 befasst sich mit ähnlichen und Verwandten Arbeiten der hier behandelten Themen. In Kapitel 4 wird der Aufbau der FMEA beschrieben sowie ein kurzer Überblick über die Zuverlässigkeit von seriellen und parallelen Systemen gegeben. Kapitel 5 befasst sich mit Continuous-time Markov chains und der Continuous Stochastic Logic, welche für die Modellierung und das Model Checking der Anlage benutzt werden. Zudem werden zwei Verfahren zur Minimierung von CTMC Modellen vorgestellt. Kapitel 6 befasst sich dann mit den Model Checking Tools PRISM und MRMC, die hier kurz vorgestellt werden. Im Anschluss wird in Kapitel 7 dann die Modellierung der Kläranlage anhand der zugrunde liegenden FMEA Analyse als CTMC beschrieben. Da eine direkte Überführung der Analyse zu einem meist sehr großen Modell führt, beschreibt Kapitel 8 verschiedene Methoden zur Minimierung der Modelle. Kapitel 9 befasst sich dann mit den Model Checking und dessen Ergebnissen auf dem zuvor entwickelten Modell und gibt zudem ein Fazit dieser Arbeit.

2 Anwendung: Kläranlage

Moderne Kläranlagen reinigen das Abwasser aus Haushalten und Industrieanlagen in einem mehrstufigen Prozess. Dieser kann grob in zwei Schritte unterteilt werden, die mechanische Vorreinigung Abschnitt 2.1 sowie die biologische Reinigung Abschnitt 2.2.

2.1 Mechanische Vorreinigung

Bei der mechanischen Vorreinigung werden verschiedene Verfahren angewandt, um grobe Schmutzteilchen und Feststoffe vom restlichen Schmutzwasser zu trennen. Im ersten Schritt dieses Verfahrens durchläuft das verunreinigte Wasser den sogenannten Rechen. Der Rechen funktioniert ähnlich wie ein Sieb und filtert zunächst sehr grobe Verunreinigungen wie Steine und Äste aber auch z. B. Hygieneartikel, die fälschlich weise in das Abwasser gelangen, aus diesem.

Im nächsten Schritt gelangt das, nun von größeren Verunreinigungen befreite, Abwasser in den Sandfang. Bei dem Sandfang, der abhängig vom Aufbau der Kläranlage oft direkt mit einem Ölfang kombiniert wird, handelt es sich um ein großes, meist längliches Becken. Das Abwasser durchfließt den Sandfang mit einer Geschwindigkeit von ungefähr 0.3 m/s. Durch diese verringerte Fließgeschwindigkeit setzen sich sowohl Sand und kleine Steine am Grund des Beckens ab, als auch Öle und Fette, die sich an der Oberfläche des Beckens sammeln. Die Verunreinigungen werden dann mithilfe von Räumschilden aus dem Abwasser entfernt.

Im dritten und letzten Schritt der mechanischen Vorreinigung, dem Vorklärbecken wird das Abwasser von festen Fäkalien getrennt. Wie bereits im Sandfang wird auch hier die Fließgeschwindigkeit des Wassers, in diesem Fall auf $1,5$ cm/s, verringert um ein Absetzen der festen Bestandteile am Beckenboden zu ermöglichen. Abbildung 2.1 zeigt

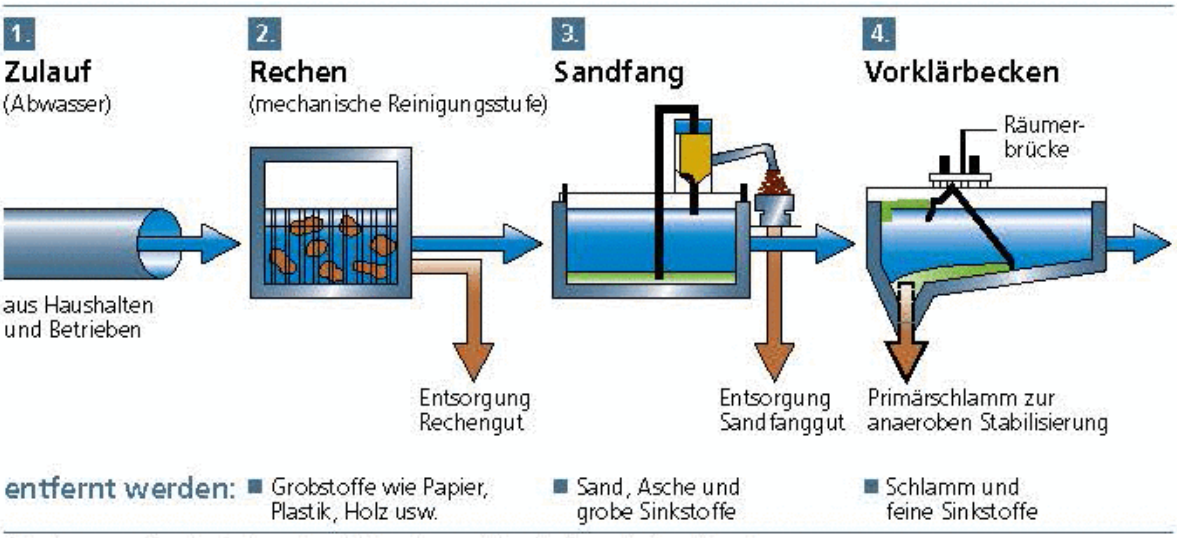

Abbildung 2.1: Schematische Darstellung der einzelnen Schritte der mechanischen Reinigung.[3]

schematisch die einzelnen Schritte der mechanischen Reinigung. Auch diese werden wieder über Räumschilde am Beckenboden vom Abwasser getrennt und anschließend verdichtet (hierzu mehr in Abschnitt 2.2).

Das nun zu größten Teilen von Feststoffen befreite Wasser ist jedoch immer noch stark verunreinigt. Das Wasser enthält eine Vielzahl von im Wasser gelösten Chemikalien, welche z. B. aus Reinigungsmitteln stammen. Diese sind auf mechanische Weise nicht zu entfernen und werden daher im nächsten Schritt, der biologischen Reinigung, entfernt.

2.2 Biologische Reinigung

Am Anfang der biologischen Reinigung steht das sogenannte Belebungsbecken. Im diesem werden dem Wasser zunächst Bakterienkulturen hinzugegeben, welche bei der Reinigung des Wassers helfen. Um die Anzahl der Bakterien schnell zu vergrößern, wird zudem der Sauerstoffgehalt des Wassers stark erhöht, was die Vermehrung der Bakterienkulturen stark beschleunigt. Dies kann auf verschiedene Weisen erfolgen. Eine verbreitete Methode ist das direkte Belüften des Wassers mithilfe von Düsen,

die sich am Grund des Beckens befinden und Luft oder reinen Sauerstoff in das Becken pumpen. Eine andere Möglichkeit die Bakterien mit ausreichend Sauerstoff zu versorgen, ist es durch "umrühren" das Wasser in dauerhafter Bewegung zu halten.

Nachdem das Wasser mit Bakterien angereichert wurde, gelangt es vom Belebungsbecken in das Nachklärbecken. Im Nachklärbecken setzen sich die zuvor im Wasser gelösten, nun jedoch durch die Bakterien in fester Form gebundenen, Verunreinigungen am Beckenboden ab. Diese werden genau so wie die Verunreinigungen im Vorklärbecken durch ein Räumschild am Beckenboden abgetragen. Eine Gefahr, die bei der Anreicherung des Wassers mit Bakterien besteht, ist die Bildung einer großen Menge von fadenförmigen Bakterien. Diese führen zur Bildung von sogenanntem Schwimmschlamm, der sich nicht am Boden des Beckens absetzt und somit nur sehr schwer vom Wasser zu trennen ist. Aus diesem Grund werden in diesem Abschnitt regelmäßig Wasserproben genommen um die Entwicklung der Bakterienkulturen genau zu überwachen.

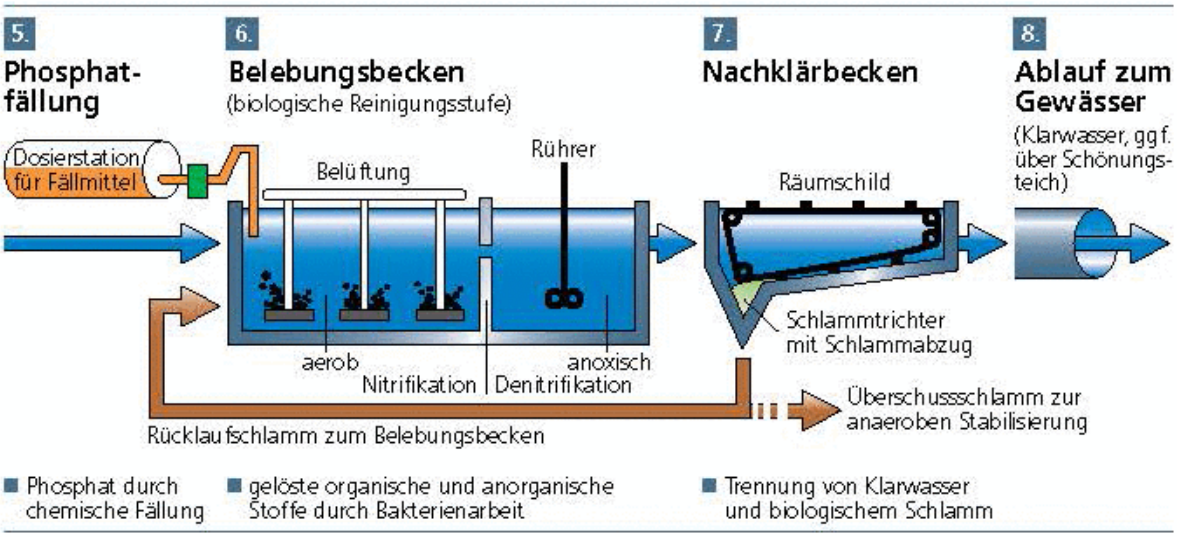

Abbildung 2.2: Schematische Darstellung der einzelnen Schritte der biologischen Reinigung.[3]

Nach diesem Reinigungsschritt befindet sich das Wasser in einem so sauberen Zustand, dass es in anliegende Flüsse geleitet werden kann. Der gesammelte Schlamm wird jedoch noch weiter verarbeitet. Zum einen wird ein Teil des bakterienhaltigen Schlammes aus dem Nachklärbecken für einen späteren Einsatz im Belebungsbecken abgetragen. Der Hauptanteil gelangt jedoch in sogenannte Verdichter. Dieser entzieht

dem Klärschlamm das überschüssige Wasser, welches sich noch in diesem befindet. Die einzelnen Schritte der biologischen Reinigung sind zudem in Abbildung 2.2 zu sehen.

Der nun verdichtete Schlamm wird abhängig vom Aufbau der Kläranlage entweder entsorgt oder gelangt in einen Faulturm. Im Faulturm kann aus dem Schlamm dann unter anderem Biogas gewonnen werden, das zur Energieversorgung der Kläranlage beiträgt.

3 Verwandte Arbeiten

Da Kläranlagen essenziell zur Erhaltung der Wasserqualität sind und somit einen Teil des Grundversorgungsnetzes darstellen, finden sie sowohl auf dem Gebiet der Sicherheits- und Zuverlässigkeitsanalyse als auch im Bereich des Model Checkings viel Beachtung. Aus diesem Grund existiert auch eine Vielzahl von Veröffentlichungen in diesem Bereich.

Zwei Veröffentlichungen, die diese Arbeit besonders beeinflusst haben, sind [6] von Hamed Ghasemieh, Boudewijn R. Haverkort und Anne Remke sowie [16] von Stephan Roolvink, Anne Remke und Mariëlle Stoelinga. [6] führt eine Analyse der Kläranlage in Enschede, welche auch dieser Arbeit als Anwendung dient, basierend auf Hybriden Petrinetzen durch. Der Fokus der Arbeit liegt hierbei auf dem Fließverhalten des Abwassers und der Aufnahmekapazität der Anlage sowie seiner einzelnen Komponenten. Zudem wurde das Verhalten der Kläranlage unter großer Belastung mithilfe der STL Logik untersucht. [16] befasst sich hingegen vorwiegend mit dem Verhalten von Wasserversorgungssystemen. Hierbei werden drei wichtige Faktoren des Systems näher untersucht, die *availability*, *reliability* und die *survivability*. Die Modellierung der Anlage wird hierbei mithilfe des Arcade tools im Form von I/O-IMCs vorgenommen, das Model Checking erfolgt mit der CSL Logik.

Des weitere existiert bereits eine Vielzahl von Fallstudien, die mit dem PRISM Model Checking Tool durchgeführt wurden. Diese befassen sich mit verschiedensten Forschungsgebieten wie, randomisierte Algorithmen, Kommunikations- und Netzwerkprotokollen, biologische Studien und viele mehr. So befasst sich [9] mit einer Fallstudie des selbst-stabilisierungs Algorithmus von Herman, welcher für Token Ringe verwendet wird. Eine andere Studie von Marta Kwiatkowska, Gethin Norman und Jeremy Sproston [10] befasst sich mit der Analyse des IEEE 1394 FireWire Protokolls und führt eine formale Verifikation des Protokolls durch. Diese Fallstudien geben

einen guten Einblick in den Aufbau einer Fallstudie, und wie dessen Ergebnisse mit PRISM verifiziert werden können, auch wenn keine direkte Verbindung zu den hier betrachteten Kläranlagen besteht.

Zudem wurden weitere Arbeiten berücksichtigt wie die Arbeit von Paul Pratheeba [15] welche sich mit der Erstellung von Reparaturplänen für Wasserversorgungsnetzwerke basierend auf stochastischen Suchalgorithmen befasst. Zudem wird eine Studie anhand des Wasserversorgungsnetzes in Chennai, Indien beschrieben, bei der mögliche Fehlerquellen und Ursachen betrachtet werden. Da auch Kläranlagen neben den verschiedenen Reinigungsbecken aus einer Vielzahl von Pumpen und Leitungen bestehen gibt diese Arbeit Anhaltspunkte über deren Verhalten. Eine weitere Veröffentlichung in diesem Bereich von M. Tabesh,J. Soltani,R. Farmani und D. Savic [19] befasst sich mit der Analyse von Leitungsversagen in Wasserversorgungsnetzen basierend auf *Artificial Neural Networks* und *Neuro-Fuzzy Systems*.

Da diese Arbeit jedoch auch auf die FMEA Analyse als Basis der Modellierung zurückgreift, wurden auch einige Veröffentlichungen aus diesem Bereich betrachtet. So befasst sich [7] mit der FMEA Analyse für eine Kläranlage, welche zur Reinigung von Wasser aus dem Tunnel- und Bergbau eingesetzt wird. Die Arbeit stellt zudem ein alternatives Modell zur FMEA Analyse basierend auf Methoden der *Fuzzy Theory* vor. Zudem wurden zwei Arbeiten [5] sowie [13] welche sich mit FMEA Analysen von Wasseraufbereitungsanlagen befassen, die eine direkte Aufbereitung von Abwasser zu neuem Trinkwasser untersuchen. Hierbei liegt der Fokus besonders auf der Verschmutzung des Wassers mit Chemikalien und Schwermetallen, die in einem geschlossenen Trinkwasserkreislauf zu Gesundheitsproblemen führen könnten.

4 Zuverlässigkeit

In diesem Kapitel wird die Failure Mode Effect Analysis (FMEA) beschrieben, dessen Ergebnisse die Grundlage der Modellierung der Kläranlage in dieser Arbeit bildet. Im Folgenden wird dann eine Erläuterung zu der Betrachtung von Zuverlässigkeit in seriellen und parallelen Systemen gegeben, welche ebenfalls im Modellierungsprozess Anwendung finden.

4.1 Failure Mode Effect Analysis (FMEA)

Die Failure Mode Effect Analysis (FMEA) ist eine Methode zur Ermittlung von Fehlern und deren Ursachen. Sie wurde erstmals von der NASA zur Qualitätssicherung der Apollo Missionen verwendet und verbreitete sich anfänglich auch in anderen Projekten der Luft- und Raumfahrt sowie der Kerntechnik. Im Laufe der Jahre fand die Methode immer mehr anklang, besonders in der Automobilindustrie, und wurde 1980 zu einer deutschen Norm der *DIN EN 60812*. Seither wird sie in immer mehr Industriezweigen angewandt.

Die Analyse erfolgt meist durch ein Team aus Experten aus allen Bereichen des Unternehmens die für die Erstellung des Systems verantwortlich sind. Hierzu zählen unter anderem Experten des Entwicklungs- und Ingenieurteams als auch Lieferanten, Produktmanager, Vertreter der Marketingabteilung sowie oftmals auch der Kunde selbst.

Um die Höhe der kosten für mögliche Änderungen so gering wie möglich zu halten sollte die FMEA so früh wie möglich in der Planungsphase eines Projekts angewandt werden. Ein Nachteil, der sich hierbei ergibt, ist jedoch, dass die Ergebnisse der FMEA um so genauer sind, um so später sie innerhalb eines Projektes durchgeführt wird, da hier bereits viele anfangs noch unklare Punkte geklärt sind, die die Durchführung

Rating	Definition
10	Fehler verursacht Personenschäden oder überschreitet Gesetzliche Vorgaben ohne Warnung
9	Fehler verursacht Personenschäden oder überschreitet Gesetzliche Vorgaben mit Warnung
8	Das System ist nicht länger funktionsfähig
7	Das System ist funktionsfähig jedoch in seiner Leistungsfähigkeit eingeschränkt
6	Das System ist funktionsfähig jedoch in Leistungsfähigkeit eines Subsystems eingeschränkt
5	Das System ist voll funktionsfähig jedoch mit einer hohen Belastung der Anwender/Betreiber verbunden
4	Der Fehler hat eine merkliche Auswirkung jedoch nur eine kleine Auswirkung auf die Leistungsfähigkeit
3	Der Fehler hat eine merkliche Auswirkung jedoch keine Auswirkung auf die Leistungsfähigkeit
2	Der Fehler hat keine Auswirkungen auf die Leistungsfähigkeit und kann von den Betreibern ignoriert werden
1	Keine signifikanten Auswirkungen

Tabelle 4.1: *Severity*(S): Schwere der Beschädigung

der FMEA erleichtern. Aus diesem Grund empfiehlt es sich im Entstehungszyklus eines Projekts mehrere FMEA Analysen an verschiedenen Zeitpunkten durchzuführen.

In vielen Fällen wird vor der Durchführung der FMEA noch eine Vorselektion durchgeführt, bei der festgelegt wird, auf welche Systemkomponenten sich die Analyse bezieht oder welche Elemente des Systems zusammengefasst werden. Dies geschieht um die Komplexität und den mit der Analyse verbundenen Zeitaufwand in moderaten Größen zu halten. Die Selektion der einzelnen Komponenten kann anhand verschiedener Merkmale erfolgen. So wird meist eine Vorauswahl anhand von kunden- oder problemorientierten Merkmalen beziehungsweise einer Kombination der beiden getroffen. Hierbei bewertet entweder der Kunde, welche der Komponenten für ihn besonders wichtig sind oder ein Team von Experten entscheidet anhand einer für das System typischen Problemstellung, welche Komponenten besonders belastbar sein müssen.

Die Methode selbst besteht aus verschiedenen Phasen der Struktur-, Funktions- und Fehleranalyse gefolgt von einer Risikobewertung und späteren Optimierung des Vorgehens. Bei der Strukturanalyse werden zunächst die einzelnen Komponenten des

Rating	Definition
10	Mehr als ein Vorfall pro Tag
9	Ein Vorfall alle drei bis vier Tage
8	Ein Vorfall pro Woche
7	Ein Vorfall pro Monat
6	Ein Vorfall alle drei Monate
5	Ein Vorfall alle sechs Monate
4	Ein Vorfall pro Jahr
3	Ein Vorfall alle ein bis drei Jahre
2	Ein Vorfall alle drei bis fünf Jahre
1	Ein Vorfall alle fünf oder mehr Jahre

Tabelle 4.2: Occurrence(O): Auftrittswahrscheinlichkeit

späteren Produkts oder der Anlage genauer untersucht und unterschieden. Abhängig vom Zeitpunkt der Analyse kann dies sowohl an ersten Konzepten, Bauplänen oder fertigen Prototypen geschehen. In der darauffolgenden Funktionsanalyse werden die Beziehungen und gegenseitigen Auswirkungen einzelner zuvor identifizierter Komponenten aufeinander identifiziert und in einer Funktionsstruktur festgehalten. Die Struktur- und Funktionsanalyse erfolgt meist sehr frei, da sie sich in verschiedenen Anwendungen stark unterscheidet. Im nächsten Schritt werden anhand der Struktur- und Funktionsanalyse mögliche Fehler identifiziert, die bei den einzelnen Komponenten beziehungsweise bei deren Zusammenspiel auftreten können. Hat sich das Team von Experten auf eine Menge zu betrachtender Fehler geeinigt, wird zudem noch deren Ursache festgestellt. Dies ermöglicht es bei einer Änderung des Systems als folge der FMEA leichter die Stellen in der Produktion oder Planung zu finden die verändert werden müssen damit das Produkt oder System den Ansprüchen der Entwickler oder des Kunden genügt.

Nachdem eine ausführliche Analyse des Systems vorliegt, wird eine Risikobewertung durchgeführt. Bei der Risikobewertung wird für jede der betrachteten Komponenten anhand der zuvor ermittelten möglichen Fehler ein Rating ermittelt. Dieses setzt sich aus drei Komponenten zusammen. So wird für jeden Fehler eine Bedeutung (*Severity S*), eine Auftrittswahrscheinlichkeit (*Occurrence O*) sowie eine Entdeckungswahrscheinlichkeit (*Detection D*) bestimmt. Jeder der drei Werte wird auf einer Skala von 1 bis 10 bewertet. Hierbei ist 1 das wünschenswerteste Ergebnis und beschreibt somit eine geringe Bedeutung oder Auftrittswahrscheinlichkeit oder eine hohe Entdeckungswahrscheinlichkeit des Fehlers, eine 10 ist somit das schlechteste Rating. Den einzelnen Zahlenwerten wird durch die Experten, die die Bewertung durchfüh-

Rating	Definition
10	Der Fehler wird oder kann nicht erkannt werden
9	Sehr entfernte Chance das der Fehler erkannt wird
8	Entfernte Chance das der Fehler erkannt wird
7	Sehr geringe Chance das der Fehler erkannt wird
6	Geringe Chance das der Fehler erkannt wird
5	Moderate Chance das der Fehler erkannt wird
4	Mäßig hohe Chance das der Fehler erkannt wird
3	Hohe Chance das der Fehler erkannt wird
2	Sehr hohe Chance das der Fehler erkannt wird
1	Der Fehler wird fast sicher erkannt

Tabelle 4.3: *Detection*(D): Detektionsrate

ren, eine jeweilig abgestufte Bedeutung zugewiesen. Dies erfolgt üblicherweise in tabellarischer Form wie in Tabelle 4.1, Tabelle 4.2 und Tabelle 4.3 zu sehen ist. Wurde jeder der Fehler durch die Experten mit den zugehörigen Ratings in den einzelnen Kategorien belegt, wird ein Gesamtrating, die sogenannte Risikoprioritätszahl (RPN) ermittelt. Hierzu werden lediglich die einzelnen Ratings der Kategorien miteinander multipliziert.

$$RPN = S \cdot O \cdot D.$$

Das Ergebnis ist eine Zahl im Wertebereich von $[1 - 1000]$, die darüber Auskunft gibt wie risikobehaftet die einzelnen Komponenten des Systems sind. Infolgedessen können dann besonders risikoreiche Komponenten entweder neu konzipiert, verbessert oder besonderen Kontrollen unterzogen werden.

Im Folgenden wird noch die Zuverlässigkeit von seriellen und parallelen Systemen besprochen. Hierbei handelt es sich nicht direkt um eine Technik zur Minimierung von Modellen, jedoch ist diese Betrachtung entscheidend, wenn ein Modell logisch abstrahiert werden soll um verschiedene Teile zusammenzufassen und zugleich den Aufbau des Modells nicht zu verfälschen.

4.2 Zuverlässigkeit von seriellen und parallelen Systemen

Bei der Modellierung komplexer Systeme wie dem einer Kläranlage müssen einige Komponenten und deren Verhalten abstrakt betrachtet werden, um die Größe des erstellten Modells auf einem Niveau zu halten, sodass übliche Model Checking Algorithmen und Datenstrukturen zur Repräsentation des Modells ausreichen. Eine grundlegende Abstraktionsebene wird bereits durch die in Abschnitt 4.1 FMEA Analyse vorgegeben. Diese wird zuvor durch das Team, dass die Analyse durchführt, bestimmt. Diese kann sehr stark variieren wie in Kapitel 3 beschrieben von den einzelnen Dichtungen und Kolben einer industriellen Pumpe bis hin zur Abstraktion ganzer Abschnitte der Kläranlage, die aus verschiedenen Pumpen und Becken bestehen.

Betrachtet man nun das System in Hinblick auf seine Zuverlässigkeit und möchte die Abstraktion einer oder mehrerer Komponenten anpassen, so muss die Zuverlässigkeit der neu erstellten Komponenten entsprechend berechnet werden. Hierzu gibt es allgemeine Berechnungsvorschriften für serielle und parallele Systeme, wie in [17] beschrieben.

Definition 1. (Analog zu [17])

Ein *serielles System* beschreibt eine Menge von Komponenten, die im Hinblick auf ihre Zuverlässigkeit seriell also in Reihe geschaltet sind. Alle Komponenten des Systems müssen funktionieren, sodass das übergeordnete System arbeiten kann. Fällt auch nur eine der Komponenten aus, so führt dies zu einem Ausfall des gesamten Systems.

Betrachtet man ein ein System von zwei unabhängigen Komponenten A und B, die im Hinblick auf ihre Zuverlässigkeit voneinander abhängig sind, so müssen beide Komponenten funktionsfähig sein, damit das gesamte System ebenfalls funktionsfähig ist.

Abbildung 4.1 zeigt dies schematisch. Seien R_A und R_B die Wahrscheinlichkeiten, dass die Komponenten A und B funktionsfähig sind sowie Q_A und Q_B die Wahrscheinlichkeit, dass die Komponenten A und B ausfallen. Da R und Q komplementäre Wahrscheinlichkeiten sind, ergibt sich:

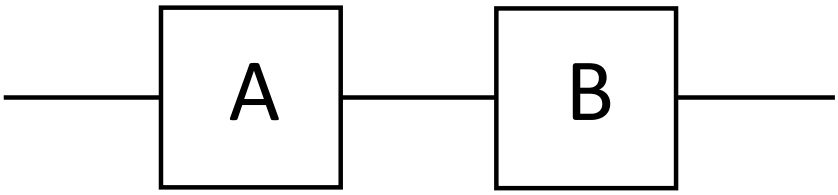

Abbildung 4.1: Ein serielles System bestehend aus zwei Komponenten A und B.

$$R_A + Q_A = 1 \text{ sowie } R_B + Q_B = 1.$$

Da das serielle System genau dann funktionsfähig ist, wenn alle Teilkomponenten funktionsfähig sind, ergibt sich für das gesamte System:

$$R_s = R_A \cdot R_B, \text{ sowie für n Komponenten } R_s = \prod_{i=1}^{n} R_i.$$

Die Wahrscheinlichkeit, dass das System einen Fehler erleidet ist somit gegeben durch:

$$\begin{aligned} Q_S &= 1 - R_A \cdot R_B \\ &= 1 - (1 - Q_a)(1 - Q_B) \\ &= Q_A + Q_B - Q_A \cdot Q_B. \end{aligned}$$

Für ein System mit n Komponenten ergibt sich:

$$Q_S = 1 - \prod_{i=1}^{n} R_i.$$

Definition 2. (Analog zu [17])

Ein *paralleles System* beschreibt eine Menge von Komponenten, die die im Hinblick auf ihre Zuverlässigkeit parallel geschaltet sind. Lediglich eine Komponente des Systems muss funktionieren, damit auch das übergeordnete System arbeiten kann. Fallen alle Komponenten aus, so führt dies auch zu einem Ausfall des gesamten Systems.

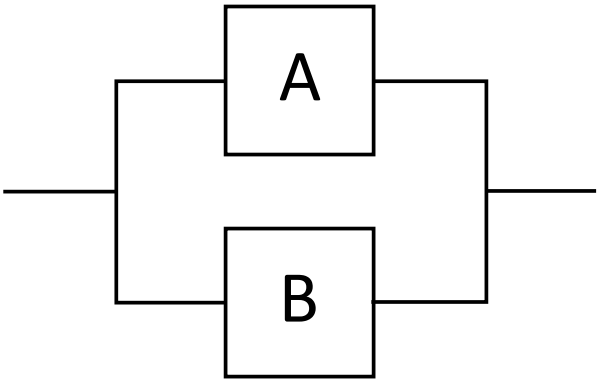

Abbildung 4.2: Ein paralleles System bestehend aus zwei Komponenten A und B.

Betrachtet man ein System von zwei parallelen Komponenten A und B wie in Abbildung 4.2 zu sehen ist, so ist zu erkennen das entweder eine der beiden Komponenten oder beide funktionsfähig sein müssen, damit das gesamte System funktionsfähig ist.

Mit einer analogen Notation ergibt sich:

$$\begin{aligned} R_P &= 1 - Q_A \cdot Q_B \\ &= R_A + R_B - R_A \cdot R_B. \end{aligned}$$

Sowie für n Komponenten:

$$R_p = 1 - \prod_{i=1}^{n} Q_i,$$

sowie:

$$Q_P = Q_A \cdot Q_B,$$

und bei einem System mit n Komponenten:

$$Q_P = \prod_{i=1}^{n} Q_i.$$

Somit ergeben sich für serielle und parallele Systeme identische Formeln wobei lediglich R und Q vertauscht ist. Dies ist auch intuitiv verständlich, da die Zuverlässigkeit eines seriellen Systems mit der Anzahl seiner Komponenten abnimmt, ein paralleles System jedoch an Zuverlässigkeit gewinnt.

5 Continuous-time Markov chains (CTMCs) und Continuous Stochastic Logic (CSL)

In diesem Kapitel werden Continuous-time Markov chains (CTMCs) sowie die Continuous Stochastic Logic (CSL) vorgestellt. Beide sind ein Teil der grundlegenden Techniken des quantitativen Model Checkings. Zudem werden zwei Methoden zur Minimierung von CTMCs die *Symmetry Reduction* und die *Bisimulation Minimization* vorgestellt, welche Anwendung in dieser Arbeit finden.

5.1 Continuous-time Markov chains (CTMCs)

Dieser Abschnitt befasst sich mit allen wichtigen Definitionen, die zum Verständnis einer CTMC benötigt werden. Hierzu wird zunächst eine DTMC definiert, bevor wir zu der Definition der CTMC übergehen. Diese lässt sich dann auf ihre eingebettete DTMC zurückführen. Zudem wird die zu einer CTMC gehörenden Generatormatrix sowie der Begriff des *Steady-State* und der *Uniformisation* beschrieben. Alle hierbei verwendeten Definitionen und Bezeichnungen dieses Abschnittes entstammen [12].

Definition 3. (Analog zu [12])

Eine *labelled* Discrete-Time Markov Chain (*DTMC*) ist definiert als ein Tupel $(S, \bar{s}, \mathbf{P}L)$ mit:

- S eine *endliche* Menge von Zuständen;
- $\bar{s} \in S$ der Startzustand der DTMC;

- $\mathbf{P} : S \times S \to [0,1]$ die Wahrscheinlichkeitsmatrix der Transitionen mit $\sum_{s' \in S} \mathbf{P}(s,s') = 1$ für alle $s \in S$;
- $L : S \to 2^{AP}$ eine Funktion, die jedem Zustand $s \in S$ eine Menge $L(s)$ von *atomaren Eigenschaften* aus der Menge der atomaren Eigenschaften AP zuweist, die für diesen Zustand zutreffen.

Hierbei ist AP eine feste und endliche Menge von atomaren Eigenschaften, die dazu verwendet wird, die einzelnen Zustände zu bezeichnen. Jedes Element der Wahrscheinlichkeitsmatrix $\mathbf{P}(s,s')$ beschreibt die jeweilige Wahrscheinlichkeit über eine Transition vom Zustand s in den Zustand s' zu gelangen. Hierbei ergibt die Summe aller ausgehenden Transpositionen eines jeden Zustands immer eins. Terminierende Zustände, von denen keine Transition zu einem anderen Zustand führen soll, können über einen *self-loop*, eine Transition, die vom Ausgangszustand wieder auf sich selbst zeigt, beschrieben werden.

Mit dieser Definition einer DTMC können wir nun auf die, sehr ähnliche, Definition einer CTMC eingehen, da diese später auf eine DTMC zurückgeführt wird.

Definition 4. (Analog zu [12])

Eine *labelled* Continuous-Time Markov Chain (*CTMC*) ist definiert als ein Tupel $\mathcal{C} = (S, \bar{s}, \mathbf{R}, L)$ mit:

- S eine *endliche* Menge von Zuständen;
- $\bar{s} \in S$ der Startzustand der CTMC;
- $\mathbf{R} : S \times S \to \mathbb{R}_{\geq 0}$ Transitions-Raten Matrix;
- $L : S \to 2^{AP}$ eine Funktion, die jedem Zustand $s \in S$ eine Menge $L(s)$ von *atomaren Eigenschaften* aus der Menge der atomaren Eigenschaften AP zuweist, die für diesen Zustand zutreffen.

Die Transitions-Raten Matrix $\mathbf{R}$ weist jeweils einem Paar von Zuständen in der CTMC eine Rate zu. Diese wird als Parameter für die Exponentialverteilung der zugehörigen Transition zwischen diesen Zuständen genutzt. Eine Transition zwischen den Zuständen s und s' besteht nur, wenn $R(s,s') > 0$ also eine Rate Größer als 0 vorliegt. Die Wahrscheinlichkeit, dass die Transition in t Zeitschritten schaltet, beträgt dann $1 - e^{-\mathbf{R}(s,s') \cdot t}$. Können von einem Zustand aus mehrere Folgezustände erreicht werden,

also existieren mehrere Zustände s' mit $\mathbf{R}(s, s') > 0$, so beschreibt dies eine *race condition* zwischen den einzelnen Transitionen. Hierbei bestimmt die erste der Transitionen die schaltet dadurch den Folgezustand in der CTMC. Die Zeit, in der sich die CTMC im Zustand s befindet, bevor eine der Transitionen schaltet, ist exponentiell verteilt mit der Rate $E(s)$. Diese ist definiert als:

$$E(s) \stackrel{\text{def}}{=} \sum_{s' \in S} \mathbf{R}(s, s').$$

$E(s)$ wird auch als *Abgangsrate* des Zustands s, also der gesamten Rate, mit der der Zustand s verlassen werden kann, bezeichnet. Ein Zustand s mit einer *Abgangsrate* von $E(s) = 0$ wird als *absorbierend* bezeichnet und besitzt somit keine ausgehenden Transitionen.

Es ist zudem möglich die Wahrscheinlichkeit zu bestimmen, mit der jeder der möglichen Folgezustände s' des Ausgangszustands s als Folgezustand gewählt wird. Hierzu wird eine DTMC benötigt.

Definition 5. (Analog zu [12])

Die *eingebettete* DTMC der CTMC $\mathcal{C} = (S, \bar{s}, \mathbf{R}, L)$ ist die DTMC $emb(\mathcal{C}) = (S, \bar{s}, \mathbf{P}^{emb(\mathcal{C})}, L)$, bei der für die Zustände $s, s' \in S$ gilt:

$$\mathbf{P}^{emb(\mathcal{C})}(s, s') = \begin{cases} \frac{R(s,s')}{E(s)} & \text{wenn } E(s) \neq 0, \\ 1 & \text{wenn } E(s) = 0 \text{ und } s = s', \\ 0 & \text{sonst.} \end{cases}$$

Mit der so definierten DTMC ist es möglich, das Verhalten der CTMC auf andere Weise zu betrachten. Die CTMC verbleibt im Zustand s mit einer exponentiell verteilten Verzögerung die der Rate $E(s)$ entspricht, bevor eine Transition schaltet. Die jeweiligen Wahrscheinlichkeiten, das der Folgezustand s' gewählt wird, ist durch $\mathbf{P}^{emb(\mathcal{C})}(s, s')$ gegeben. Zur späteren Analyse der CTMC wird zudem noch die folgende Matrix benötigt.

Definition 6. (Analog zu [12])

Die *infinitesimale genarator Matrix* für die CTMC $\mathcal{C} = (S, \bar{s}, \mathbf{R}, L)$ ist die Matrix $\mathbf{Q} : S \times S \to \mathbb{R}$ definiert als:

$$\mathbf{Q}(s,s') = \begin{cases} \mathbf{R}(s,s') & \text{wenn } s \neq s', \\ -\sum_{s'' \neq s} \mathbf{R}(s,s'') & \text{sonst.} \end{cases}$$

Beispiel 1. In Abbildung 5.1 ist ein Beispiel der CTMC $\mathcal{C}_\infty = (S_1.\bar{s}_1, \mathbf{R_1}, L)$ zu sehen. Die einzelnen Zustände werden durch Kreise dargestellt, die mit den zugehörigen Zuständen $S_1 = \{s_0, s_1, s_2\}$ gekennzeichnet sind. Die Transitionen zwischen den Zuständen werden durch Pfeile realisiert. Diese werden mit ihren zugehörigen Raten beschriftet. Der Startzustand der CTMC $\bar{s}_1 = s_0$ wird zusätzlich durch einen Pfeil markiert, der auf diesen zeigt, jedoch von keinem anderen Zustand ausgeht. Zudem können die einzelnen Zustände der CTMC mit Bezeichnungen aus der Menge $AP = \{\text{Normalberieb}, \text{Ausgefallen}, \text{Reparieren}\}$ beschriftet sein. Hierbei kann jeder der Zustände mit einem, keinen oder mehreren Bezeichnungen beschriftet sein.

Da das hier verwendete Beispiel die Modellierung einer Pumpe darstellt, besitzt jeder der Zustände eine einzige Bezeichnung, die jeweils den Zustand beschreibt, in dem sich die Pumpe zu einem bestimmten Zeitpunkt befindet. Sie kann entweder regulär funktionieren ($\{Normalbetrieb\}$), durch eine Beschädigung nicht länger funktionstüchtig sein ($\{Ausgefallen\}$) oder der Funktionsfehler wurde bereits bemerkt und die Pumpe wird repariert ($\{Reparieren\}$). Die Raten der Transitionen beschreiben in diesem Modell den Zeitraum, der durchschnittlich vergeht, bis ein Ereignis eintritt. Hierbei entspricht eine Rate von 1 jeweils einem Tag. Eine Pumpe erleidet durchschnittlich alle 6 Monate einen Ausfall ($\frac{1}{180}$). Ist die Pumpe beschädigt, wird dies durchschnittlich nach zwölf Stunden erkannt (2) und innerhalb von zwei Stunden repariert (12). Die zugehörige Matrix der Transitionsraten $\mathbf{R}_1$, die Wahrscheinlichkeitsmatrix $\mathbf{P}_1^{emb(\mathcal{C}_1)}$ und die infinitesimale genarator Matrix $\mathbf{Q}_1$ sehen wie folgt aus:

$$\mathbf{R}_1 = \begin{pmatrix} 0 & \frac{1}{180} & 0 \\ 0 & 0 & 2 \\ 12 & 0 & 0 \end{pmatrix}, \quad \mathbf{P}_1^{emb(\mathcal{C}_1)} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad \mathbf{Q}_1 = \begin{pmatrix} -\frac{1}{180} & \frac{1}{180} & 0 \\ 0 & -2 & 2 \\ 12 & 0 & -12 \end{pmatrix}.$$

Im Folgenden muss noch definiert werden wie Pfade und deren Wahrscheinlichkeiten in CTMCs bestimmt werden können, da dies unter anderem Grundlage für die spätere Definition des *Steady-States* ist und zudem im späteren Model Checking benötigt wird.

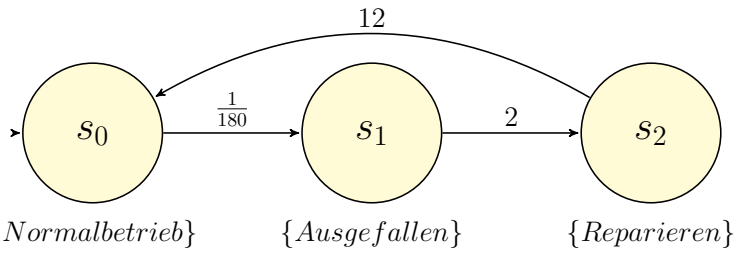

Abbildung 5.1: Beispiel der CTMC $\mathcal{C}_1$ anhand einer Klärwerkspumpe.

(Analog zu [12]). Ein unendlicher Pfad durch eine CTMC $\mathcal{C} = (S, \bar{s}, \mathbf{R}, L)$ ist eine nicht leere Sequenz $s_0 t_0 s_1 s_2 \ldots$ wobei $\mathbf{R}(s_i, s_{i+1} > 0$ ist und $t_i \in \mathbb{R} > 0$ für alle $i \geq 0$. Ein endlicher Pfad ist somit eine Sequenz $s_0 t_0 s_1 t_1 s_{k-1} t_{k_1} s_k$, wobei s_k einen absorbierenden Zustand darstellt. Hierbei gibt der Wert t_i die Zeit an, die die CTMC im Zustand s_i verbracht hat. Ein Pfad wird zudem als ω bezeichnet. Der $i$te Zustand s_i wird mit $\omega(i)$ bezeichnet. Die Bezeichnung $time(\omega, i)$ beschreibt für einen unendlichen Pfad ω die Zeit, die in Zustand s_i verbracht wird, also t_i. Die Bezeichnung $\omega@t$ beschreibt den Zustand, in welchem sich die CTMC zum Zeitpunkt t befindet.

Für einen endlichen Pfad $\omega = s_0 t_0 s_1 s_2 \ldots t_{k-1} s_k$, ist $time(\omega, i)$ lediglich für $i \leq k$ definiert, mit $time(\omega, j) = t_i$ für $i < k$ und $time(\omega, k) = \infty$. Zudem ist für $t \leq sum_{i_0}^{k-1} t_i$ $\omega@t$ definiert als unendlicher Pfad und andernfalls gilt $\omega@t = s_k$. Die Menge aller Pfade der CTMC $\mathcal{C}$ wird bezeichnet mit $Path^{\rfloor}(s)$ mit dem Startzustand s.

Mit den nun beschriebenen Bezeichnungen für die Pfade der CTMC können wir das *transiente Verhalten*, welches den Zustand der CTMC zu einem bestimmten Zeitpunkt beschreibt, sowie den *Steady-State*, welcher das Verhalten der CTMC über einen längeren Zeitraum hin beschreibt. einer CTMC definieren.

(Analog zu [12]). Für eine CTMC $\mathcal{C} = (S, \bar{s}, \mathbf{R}, L)$ beschreibt die transiente Wahrscheinlichkeit $\pi^{\mathcal{C}}_{s,t}(s')$ die Wahrscheinlichkeit sich zum Zeitpunkt t im Zustand s' zu befinden mit dem Startzustand s. Sie ist somit definiert als:

$$\pi^{\mathcal{C}}_{s,t}(s') \stackrel{\text{def}}{=} Pr_s\{\omega \in Path^{\mathcal{C}}(s) \mid \omega@t = s'\}.$$

Die Steady-State Wahrscheinlichkeit kann dann wie folgt definiert werden:

$$\pi_s^{\mathcal{C}}(s') \stackrel{\text{def}}{=} lim_{t\to\infty}\, \pi_{s,t}^{\mathcal{C}}(s').$$

Mit der so bestimmten Wahrscheinlichkeitsverteilung des Steady-State $\pi_s^{\mathcal{C}}(s')$ für alle $s' \in S$ lässt sich dann prozentual bestimmen, wie viel Zeit die CTMC in einem jeden Zustand verbringt. Für alle CTMCs mit einem endlichen Zustandsraum existiert der Limes des Steady-States immer [18]. Zudem ist für eine *irreduzible* CTMC, also eine CTMC, bei der jeder Zustand von jedem anderen aus erreicht werden kann, die Steady-State Wahrscheinlichkeit $\pi_s^{\mathcal{C}}(s')$ unabhängig vom Startzustand s der CTMC.

Eine sehr verbreitete Technik zur Berechnung von transienten Wahrscheinlichkeiten in CTMCs ist die sogenannte *uniformisation*.

Uniformisation. (Analog zu [12])

Für eine CTMC $\mathcal{C} = (S, \bar{s}, \mathbf{R}, L)$ wird mit $\Pi_t^{\mathcal{C}}$ die Matrix aller transienten Wahrscheinlichkeiten zum Zeitpunkt t beschrieben. z. B. $\Pi_t^{\mathcal{C}}(s, s') = \pi_{s,t}^{\mathcal{C}}(s')$. Es kann gezeigt werden, dass $\Pi_t^{\mathcal{C}}$ als ein Matrixexponent ausgedrückt werden kann:

$$\Pi_t^{\mathcal{C}} = e^{\mathbf{Q}\cdot t} = \sum_{i=0}^{\infty} \frac{(\mathbf{Q}\cdot t)^i}{i!}.$$

Hierbei beschreibt $\mathbf{Q}$ die infinitesimale generator Matrix von $\mathcal{C}$. Da diese Weise der Berechnung jedoch instabil ist, wird die Berechnung stattdessen auf der *uniformised* DTMC von $\mathcal{C}$ berechnet.

Definition 7. (Analog zu [12])

Für eine beliebige CTMC $\mathcal{C} = (S, \bar{s}, \mathbf{R}, L)$ mit der infinitesimalen generator Matrix $\mathbf{Q}$ ist die *uniformised* DTMC gegeben durch $\texttt{unif}(\mathcal{C} = (S, \bar{s}, \mathbf{P}^{\texttt{unif}(\mathcal{C})}, L)$ wobei $\mathbf{P}^{\texttt{unif}(\mathcal{C})} = \mathbf{I} + \frac{\mathbf{Q}}{q}$ mit einem $q \geq max\{E(s) \mid s \in S\}$.

Mit q wir die sogenannte *uniformisation rate* beschrieben. Diese wird durch den Zustand bestimmt, der die geringste durchschnittliche Wartezeit besitzt. Mit dieser Rate q werden alle Wartezeiten der CTMC $\mathcal{C}$ dann normalisiert. Dies bedeutet, dass für jeden Zustand $s \in S$ mit $E(s) = q$ jeweils ein Zeitraum in $\texttt{unif}(\mathcal{C})$ einer einzelnen exponentiell verteilten Verzögerung mit Rate q entspricht. Daraufhin wird einer der Folgezustände entsprechend der Verteilung gewählt. Somit werden keine *self loops* in

der DTMC $\texttt{unif}(\mathcal{C})$ hinzugefügt. Besitzt ein Zustand eine längere durchschnittliche Wartezeit als $\frac{1}{q}$ also $E(s) < q$ so besteht die Möglichkeit, dass einer der Zeiträume in $\texttt{unif}(\mathcal{C})$ nicht genügt um zu einem Folgezustand zu gelangen. Daher werden solche Zustände mit einem *self loop* mit Wahrscheinlichkeit $1 - \frac{E(s)}{q}$ versehen. Mithilfe der *uniformised* DTMC ist es nun möglich die Matrix der Transitionswahrscheinlichkeiten, wie flogt auszudrücken:

$$\Pi_t^{\mathcal{C}} = \sum_{i=0}^{\infty} \gamma_{i,q\cdot t} \cdot \left(\mathbf{P}^{\texttt{unif}(\mathcal{C})}\right)^i \text{ mit } \gamma_{i,q\cdot t} = e^{-q\cdot t} \cdot \frac{(q\cdot t)^i}{i!}.$$

Hierbei beschreibt jeder Schritt in der uniformised DTMC einer exponentiell verteilten Verzögerung mit dem Parameter q. Die Potenzmatrix $(P^{\texttt{unif}(\mathcal{C})})^i$ beschreibt die Wahrscheinlichkeit eines Übergangs zwischen je zwei Zuständen der DTMC in i Schritten. Zudem beschreibt $\gamma_{i,q\cdot t}$ die $i$te Poisson Wahrscheinlichkeit mit dem Parameter $q \cdot t$, dass i Schritte in t Zeitschritten begangen werden mit einer exponentiell verteilen Verzögerung q.

Zudem ist es möglich die Berechnung effektiv als Matrix-Vektor Multiplikation, anstatt als Matrix-Matrix Multiplikation durchzuführen. Betrachtet man die die Berechnung von $\pi_{s,t}^{\mathcal{C}}(s')$ für einen fixen Zustand s, so ist es möglich diese Werte durch die vorherige Multiplikation der Matrix $\mathbf{\Pi}_t^{\mathcal{C}}$ mit der initialen Wahrscheinlichkeitsverteilung zu erhalten. In diesem Fall die Multiplikation des Vektors $\underline{\pi}_{s,0}^{\mathcal{C}}$ wobei $\pi_{s,0}^{\mathcal{C}}$ den Wert 1 besitzt für $s' = s$ und andernfalls 0. Wir erhalten:

$$\underline{\pi}_{s,t}^{\mathcal{C}} = \underline{\pi}_{s,0}^{\mathcal{C}} \cdot \mathbf{\Pi}_t^{\mathcal{C}} = \underline{\pi}_{s,0}^{\mathcal{C}} \cdot \sum_{i=0}^{\infty} \gamma_{i,q\cdot t} \cdot \left(\mathbf{P}^{\texttt{unif}(\mathcal{C})}\right)^i.$$

Durch einige Umformungen kann dies als Summe von Vektoren dargestellt werden:

$$\underline{\pi}_{s,t}^{\mathcal{C}} = \sum_{i=0}^{\infty} \left(\gamma_{i,q\cdot t} \cdot \underline{\pi}_{s,0}^{\mathcal{C}} \cdot \left(\mathbf{P}^{\texttt{unif}(\mathcal{C})}\right)^i\right).$$

Hierbei kann jeder Vektor durch eine Matrix-Vektor Multiplikation mit dem Vektor der vorherigen Iteration bestimmt werden.

$$\underline{\pi}_{s,0}^{\mathcal{C}} \cdot \left(\mathbf{P}^{\texttt{unif}(\mathcal{C})}\right)^i = \left(\underline{\pi}_{s,0}^{\mathcal{C}} \cdot \left(\mathbf{P}^{\texttt{unif}(\mathcal{C})}\right)^{i-1}\right) \cdot \mathbf{P}^{\texttt{unif}(\mathcal{C})}.$$

Ein weiterer Vorteil dieser Betrachtungsweise ist, dass die Berechnung schrittweise erfolgen kann und es möglich ist, die Größe des bestehenden Fehlers nach einer bestimmten Anzahl von Summationsschritten bereits vor der Berechnung dieser zu bestimmen. Dies ist wie folgt möglich:

$$\sum_{i=0}^{\infty} e^{-q\cdot t} \cdot \frac{(q\cdot t)^i}{i!} \cdot \underline{\pi}^{\mathcal{C}}_{s,0} \cdot \left(\mathbf{P}^{\mathtt{unif}(\mathcal{C})}\right)^i - \sum_{i=0}^{k_\epsilon} e^{-q\cdot t} \cdot \frac{(q\cdot t)^i}{i!} \cdot \underline{\pi}^{\mathcal{C}}_{s,0} \cdot \left(\mathbf{P}^{\mathtt{unif}(\mathcal{C})}\right)^i$$
$$= \sum_{i=k_\epsilon+1}^{\infty} e^{-q\cdot t} \cdot \frac{(q\cdot t)^i}{i!} \cdot \underbrace{\underline{\pi}^{\mathcal{C}}_{s,0} \cdot \left(\mathbf{P}^{\mathtt{unif}(\mathcal{C})}\right)^i}_{\leq 1}.$$

Hierbei beschreibt k_ϵ die Anzahl der Schritte die benötigt werden um einen maximalen Fehler von ϵ zu erreichen sowie ϵ die Größe des Fehlers. Der Fehler lässt sich nin wür ein gewähltes k_ϵ wie folgt abschätzen:

$$\sum_{i=0}^{\infty} e^{-q\cdot t} \cdot \frac{(q\cdot t)^i}{i!} = 1 - \sum_{i=0}^{k_\epsilon} e^{-q\cdot t} \cdot \frac{(q\cdot t)^i}{i!} \leq \epsilon.$$

Somit ist es möglich die Berechnung der Uniformisation effektiv und mit einer beliebigen Genauigkeit durchzuführen.

Im Folgenden betrachten wir erneut die in Abbildung 5.1 beschriebene CTMC als Beispiel für die Durchführung einer Uniformisation.

Beispiel 2. Wie zuvor beschrieben lässt sich die Matrix $\mathbf{P}^{\mathtt{unif}}(\mathcal{C})$ wie folgt bestimmen:

$$\mathbf{P}^{\mathtt{unif}}(\mathcal{C}) = \mathbf{I} + \frac{\mathbf{Q}}{q}.$$

Wie bereits aus dem vorherigen Beispiel bekannt sehen die Matrix $\mathbf{Q}$ wie folgt aus:

$$\mathbf{Q} = \begin{pmatrix} -\frac{1}{180} & \frac{1}{180} & 0 \\ 0 & -2 & 2 \\ 12 & 0 & -12 \end{pmatrix}.$$

Es lässt sich nun leicht erkennen, dass $q = 12$ ein valider Wert für dessen Wahl ist da $q \geq max\{E(s) \mid s \in S\}$. Es ergibt sich somit:

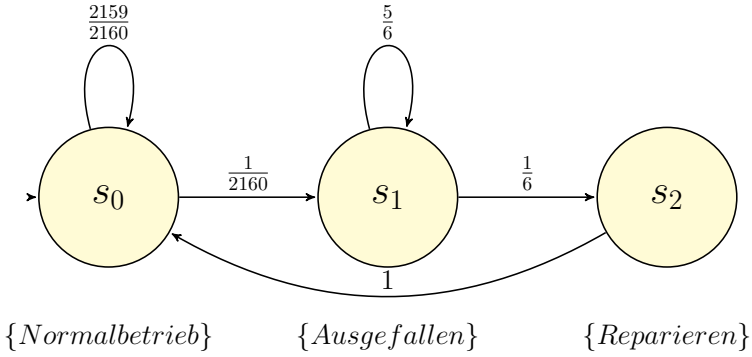

Abbildung 5.2: Abbildung der *uniformised* DTMC der Klärwerkspumpe.

$$\mathbf{P}^{\mathtt{unif}}(\mathcal{C}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} -\frac{1}{2160} & \frac{1}{2160} & 0 \\ 0 & -\frac{1}{6} & \frac{1}{6} \\ 1 & 0 & -1 \end{pmatrix} = \begin{pmatrix} \frac{2159}{2160} & \frac{1}{2160} & 0 \\ 0 & \frac{5}{6} & \frac{1}{6} \\ 1 & 0 & 0 \end{pmatrix}.$$

Die *uniformised* DTMC entspricht somit der in Abbildung 5.2 zu sehenden DTMC.

Wählen wir nun als Ausgangsvektor $\underline{\pi}^{\mathcal{C}}_{s,0} = (1,0,0)$ um die transienten Wahrscheinlichkeiten zum Zeitpunkt $t = 1$ so lässt sich die transiente Wahrscheinlichkeit wie folgt berechnen:

$$\underline{\pi}^{\mathcal{C}}_{s,1} = \sum_{i=0}^{\infty} e^{-12\cdot 1} \frac{(12\cdot 1)^i}{i!} \cdot (1,0,0) \cdot \begin{pmatrix} \frac{2159}{2160} & \frac{1}{2160} & 0 \\ 0 & \frac{5}{6} & \frac{1}{6} \\ 1 & 0 & 0 \end{pmatrix}^i.$$

Da die unendliche Summe nicht ohne weiteres berechnet werden kann muss nun eine Abschätzung der Schrittanzahl für eine gegebenen Fehlergenauigkeit erfolgen. In diesem Fall erlauben wir beispielsweise einen Fehler von $\epsilon = 0.0005$. Die Summation der Fehlerabschätzung lässt sich wie bereits beschrieben wie folgt bestimmen:

$$1 - \sum_{i=0}^{k_\epsilon} e^{-q\cdot t} \cdot \frac{(q\cdot t)^i}{i!} \leq \epsilon.$$

Es ergibt sich:

$$1 - \sum_{i=0}^{24} e^{-12\cdot 1} \cdot \frac{(12\cdot 1)^i}{i!} \approx 0.000685 \not\leq 0.0005,$$
$$1 - \sum_{i=0}^{25} e^{-12\cdot 1} \cdot \frac{(12\cdot 1)^i}{i!} \approx 0.000307 \leq 0.0005.$$

Somit werden 25 Summationen benötigt, sodass der entstehende Fehler den gewählten Wert $\epsilon = 0.0005$ nicht überschreitet. Somit erhalten wir folgendes Ergebnis für die transienten Wahrscheinlichkeiten:

$$\underline{\pi}^{\mathcal{C}}_{s,1} = \sum_{i=0}^{25} e^{-12\cdot 1} \frac{(12\cdot 1)^i}{i!} \cdot (1,0,0) \cdot \begin{pmatrix} \frac{2159}{2160} & \frac{1}{2160} & 0 \\ 0 & \frac{5}{6} & \frac{1}{6} \\ 1 & 0 & 0 \end{pmatrix}^i \approx (0.996909, 0.002395, 0.000386).$$

Über diese Form der Berechnung lassen sich die transienten Wahrscheinlichkeiten der einzelnen Zustände leicht bestimmen. Dieses Verfahren wird unter anderem für die Berechnung der später beschriebenen Until-Formeln der CSL Logik verwendet.

Parallele Komposition. Eine weitere Technik, die in dieser Arbeit Anwendung findet, ist die *parallele Komposition* von CTMCs. Hierbei werden zwei oder mehrere CTMCs zu einer neuen CTMC verbunden, wobei jeder Zustand der neuen CTMC jeweils eine Kombination von Zuständen aller CTMCs beschreibt, aus der die Komposition entstanden ist. Hierbei können zudem Zustände verschiedener CTMCs zusammengefasst werden, wenn diese mit einem gleichen Label versehen sind. Dies wird als *Synchronisation* bezeichnet. Die Erstellung einer parallelen Komposition ist auch in PRISM möglich, hierbei werden die einzelnen Teilmodelle als *Module* bezeichnet, aus denen dann das Modell zusammengesetzt wird. Mehr hierzu in Kapitel 6. Zur besseren Verdeutlichung des Prinzips folgt nun ein Beispiel für eine parallele Komposition.

Beispiel 3. In diesem Beispiel betrachten wir erneut die Modelle zweier Pumpen wie sie bereits in vorherigen Beispielen beschrieben wurden. Abbildung 5.3 zeigt links beide Pumpen in ihrem Ausgangszustand.

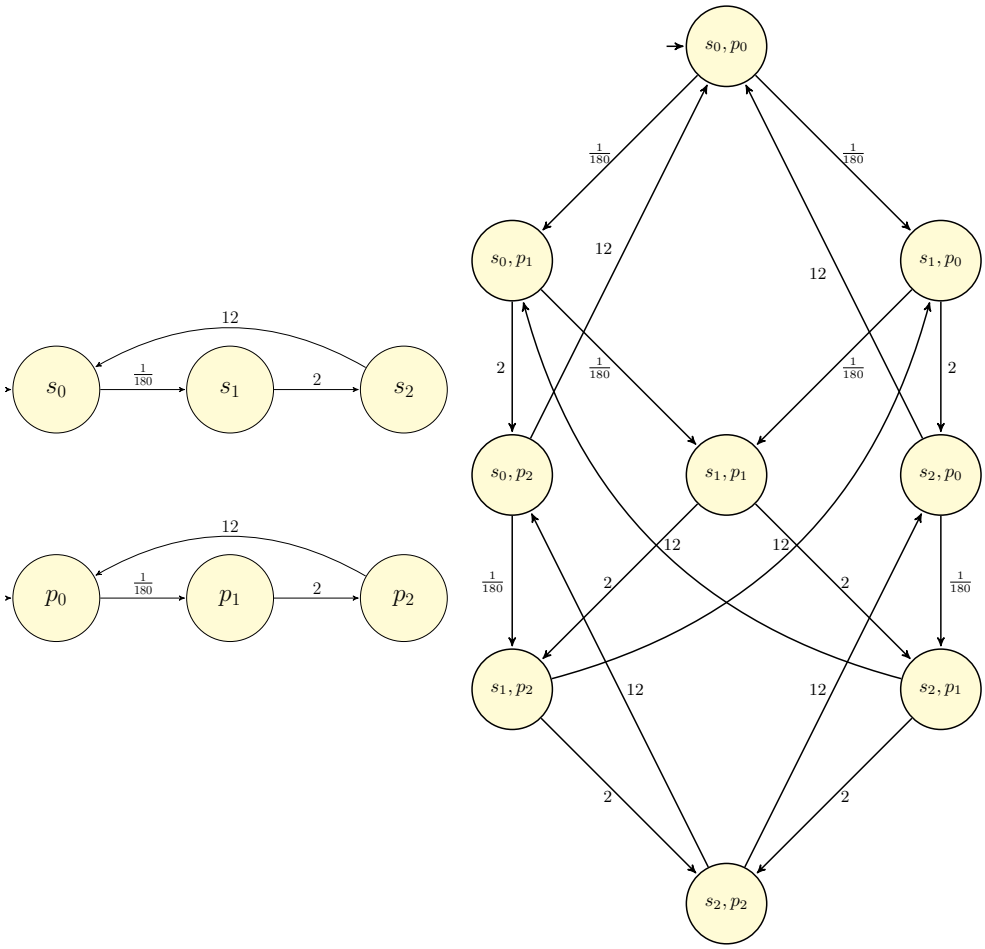

Abbildung 5.3: Beispiel einer parallelen Komposition von zwei Pumpen s und p. (Links) die beiden pumpen separat. (Rechts) Parallele Komposition der beiden Pumpen.

Die Zustände der oberen Pumpe werden mit s_0, s_1 und s_2 bezeichnet wohingegen die Zustände der unteren mit p_0, p_1 und p_2 bezeichnet werden. Die parallele Komposition, die sich aus den beiden Pumpen ergibt ist in der Abbildung rechts zu sehen. Wie bereits zuvor beschrieben, werden jeweils zwei der Zustände der zugrunde liegenden CTMCs miteinander zu einem neuen Zustand verbunden. Da der Ausgangszustand unserer Pumpten s_0 und p_0 sind entsteht ein neuer Ausgangszustand s_0, p_0. Von diesem aus müssen nun Zustände erreichbar sein, in denen s_1 und p_1 enthalten sind da dies die Folgezustände von s_0 und p_0 sind. Bei den Übergängen ßchaltet"jeweils nur immer eine der beiden zugrundeliegenden CTMCs, nie beide gleichzeitig. Die Folgezustände sind somit s_0, p_1 und s_1, p_0. Auch die Raten der Transitionen sind identisch zu denen der beiden CTMCs aus denen die parallele Komposition gebildet wurde.

Auf diese Weise ist es nun möglich, das Verhalten mehrerer Komponenten gleichzeitig zu betrachten. Das aus der parallelen Komposition der beiden CTMCs entstandene Modell ist wieder eine CTMC und auch die Schaltweise der Transitionen wird immer noch über eine *race condition* abgebildet [1].

5.2 Minimierung von CTMCs

Die Größe von Modellen stellt oft ein Problem beim Modell Checking dar. Dies beruht darauf, dass die modellierten Protokolle, Arbeitsprozesse oder mechanische Abläufe oft sehr komplex sind und nicht zu stark abstrahiert werden können, wenn zugleich signifikante Aussagen durch das Model Checking erzielt werden sollen. Da auch die in dieser Arbeit modellierte Kläranlage eine sehr komplexe Anwendung darstellt, werden an dieser Stelle grundlegende Techniken zur Minimierung von CTMC Modellen beschrieben, welche bei der späteren Modellierung der Kläranlage Anwendung finden.

Zunächst wird die Definition einer Äquivalenzrelation zwischen Zuständen einer CTMC gegeben, da diese sowohl for dieSymmetry Reduction als auch die Bisimulation Minimization benötigt wird. Daraufhin werden die beiden Verfahren, *Symmetry Reduction* und *Bisimulation Minimization*, näher erläutert. Daraufhin folgt ein kurzer Abschnitt über die Zuverlässigkeit von seriellen und parallelen Systemen, da diese ebenfalls für die Minimierung des später beschriebenen Modells verwendet wurden.

Nun zu der Definition einer Äquivalenzrelation.

Definition 8. Eine Äquivalenzrelation R auf einer Menge von Zuständen S ist definiert als eine Relation $R \subseteq S \times S$. Die Notation $s_1\ R\ s_2$ wird als Kurzform für $(s_1.s_2) \in R$ verwendet. Hierbei ist R eine Äquivalenzrelation genau dann wenn folgende Eigenschaften erfüllt sind:

- R ist reflexiv, $s\ R\ s$;
- R ist symmetrisch, wenn $s_1\ R\ s_2$ in R enthalten ist, dann auch $s_2\ R\ s_1$;
- R ist transitiv, wenn $s_1\ R\ s_2$ und $s_2\ R\ s_3$ in R enthalten ist, dann auch $s_1\ R\ s_3$.

Die si entstehenden Äquivalenzklassen werden dann mit $[s]_R = \{s' \in S \mid s' \; R \; s\}$ bezeichnet und der Quotient von S unter R mit $S/R = \{[s]_R \mid s \in S\}$.

5.2.1 Symmetry Reduction

Die *Symmetry Reduction* ist eine Methode, die es erlaubt, den Zustandsraum von Modellen wie CTMCs zu verkleinern. Hierzu werden, wie der Name bereits besagt, symmetrische Zustände identifiziert und dann zusammengefasst. Damit dies möglich ist muss zunächst bestimmt werden, wie die Symmetry zwischen zwei Zuständen einer CTMC definiert wird.

Definition 9. (Analog zu [11])

Die Symmetry von Zuständen einer CTMc ist definiert über eine Permutation $\pi : S \to S$ auf dem Zustandsraum der CTMC. Somit muss für symmetrische Zustände gelten, dass $R(\pi(s), \pi(s')) = R(s, s')$ für alle $s, s' \in S$. Über diese Relation lassen sich die Zustände in Gruppen von Zuständen unterteilen, die sich in einer gleichen Äquivalenzklasse befinden. Definieren wir nun einen reduzierten Zustandsraum $\overline{S}$ welcher für jede der Gruppen einen repräsentativen Zustand enthält, sowie eine Funktion $rep : S \to \overline{S}$ welche die Repräsentation berechnet, lässt sich die reduzierte CTMC wie folgt konstruieren. Für eine CTMC $\mathcal{C} = (S, \mathbf{s}, \mathbf{R})$ ist das Quotienten Modell gegeben durch $(\overline{S}, \overline{\mathbf{s}}, \overline{\mathbf{R}})$ mit den Zuständen $\overline{s}, \overline{s'} \in \overline{S}$ wobei:

$$\overline{R}(\overline{s}, \overline{s'}) = \sum_{\{s' \in S \mid rep(s') = \overline{s'}\}} R(\overline{s}, s').$$

Besonders für CTMC Modelle, die aus verschiedenen identischen Komponenten über eine parallele Komposition entstanden sind ist dies eine effektive Methode den Zustandsraum zu verkleinern. Zum besseren Verständnis des Verfahrens nun ein Beispiel.

Beispiel 4. Als Beispiel für die Symmetry Reduction betrachten wir nun das Modell der zwei Pumpen, welches bereits in Abbildung 5.3 zur Veranschaulichung der parallelen Komposition diente. Betrachten wir das Ergebnis der parallelen Komposition, so ist leicht zu erkennen, dass die entstandene CTMC symmetrische Zustände besitzt. Zum

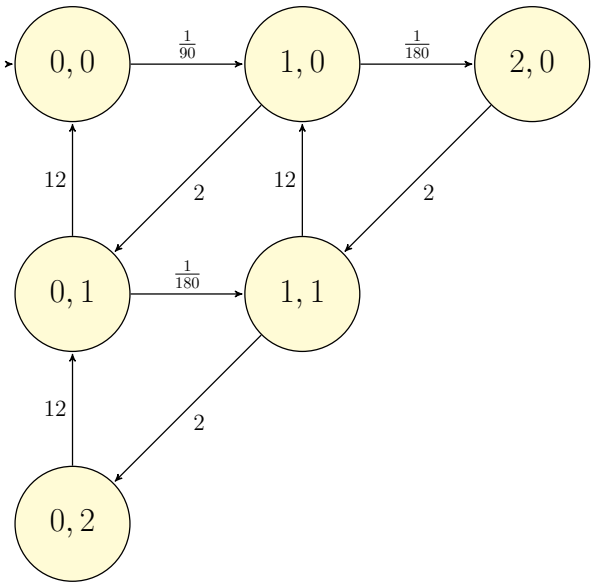

Abbildung 5.4: Beispiel der Symmetry Reduction angewannt auf die parallele Komposition zweier Pumpen aus Abbildung 5.3.

Beispiel sind die Zustände s_0, p_1 sowie s_1, p_0 symmetrisch bezüglich ihres Verhaltens gegenüber den Zuständen s_0, p_0 sowie s_1, p_1 sowie der Zustände s_0, p_2 und s_2, p_0 wenn man letztere beiden ebenfalls in eine Äquivalenzklasse zusammenfügt.

Das ganze wird etwas anschaulicher, wenn man die CTMC wieder als ein Konstrukt aus zwei Teilsystemen, in diesem Fall Pumpen, betrachtet, die jedoch nun nicht mehr von einander unterschieden werden können. Gab es im Ausgangsmodell die Zustände s_0, p_1 und s_1, p_0 hatten diese die jeweilige Aussage "Die Pumpe S (bzw. P) ist beschädigt während die Pumpe P (bzw. S) intakt ist". Diese Aussage wird nun zu der Aussage Ëine der beiden Pumpen ist beschädigt"vereinfacht. Das Ergebnis der Symmetry Reduction angewendet auf das zwei Pumpen Modell ist in Abbildung 5.4 zu sehen. Die einzelnen Zustände der CTMC sind jeweils mit einem Tupel von Zahlen versehen, dieses gibt an wie viele der Pumpen beschädigt und wie viele Pumpen in Reparatur sind, in der Form `<beschädigt>,<reparieren>`. Hierbei ist zu beachten, dass eine Pumpe immer nur entweder beschädigt oder als in Reparatur sein kann, eine Pumpe die Repariert wird ist somit implizit auch beschädigt, wird jedoch zur Vereinfachung nicht mehr in der Anzahl der beschädigten Pumpen geführt. Eine Verallgemeinerung dieses Aufbaus ist zudem in Abbildung 8.2 zu sehen.

5.2.2 Bisimulation Minimization

Eine weitere Methode den Zustandsraum eines CTMC Modells zu verkleinern ist die *Bisimulation Minimization* [8] welche wie folgt definiert ist.

Definition 10. (Analog zu [8])

Sei $\mathcal{C} = (S, \bar{s}, \mathbf{R}, L)$ eine CTMC und R eine Äquivalenzrelation auf der Zustandsmenge S. Dann ist R eine *starke Bisimulation* auf $\mathcal{C}$ wenn für die Relation $s_1 \; R \; s_2$ gilt:

- $L(s_1) = L(s_2)$;
- $\mathbf{R}(s_1, T) = \mathbf{R}(s_2, T)$ für alle T in S/R.

Es lässt sich zudem zeigen, dass die durch die Bisimulation entstandene CTMCs unter allen CSl Formeln identische Ergebnisse liefern, wie auf der ursprünglichen CTMC [4]. Um nun die Bisimulation Minimisation durchzuführen, wird ein Algorithmus benötigt, der die Bisimulation so auf die CTMC anwendet, dass eine minimierte CTMC entsteht. EIn sehr bekannter Algorithmus hierfür ist das sogenannte *partition refinement*. Der Aufbau des Algorthmus basiert auf sogenannten *Splittern* und lässt sich wie folgt skizzieren:

Partition refinement. Sei $\Pi = \{\{s \in S \mid L(s) = lab\} \mid lab \in 2^{AP}\}$ eine initiale Unterteilung von S. Ein Splitter $T \in \Pi$ für einen Block $B \in \Pi$ ist ein Block mit der Eigenschaft, dass wie Wahrscheinlichkeit den Block T zu betreten nicht die gleiche ist für alle Zustände $s \in B$. Somit $\exists s, s' \in B$ mit $R(s, T) \neq P(s', T)$. Mit den nun getroffenen Definitionen lässt sich der Algorithmus nun leicht umreisen:

- Eine Initiale Unterteilung Π von S wird gewählt.
- Es wird für jeden der Blöcke ein Splitter $T \in \Pi$ gesucht.
- Ist ein Splitter gefunden, wird der Block B in zwei Teilblöcke unterteilt, sodass $R(s, T)$ für alle Zustände des Teilblocks gleich ist.
- Dieser Vorgang wird so lange wiederholt, bis kein weiterer Splitter gefunden werden kann.

Über dieses Verfahren ist es möglich die Größe eine CTMC zu minimieren.

5.3 Continuous Stochastic Logic (CSL)

Diese Kapitel befasst sich mit der Definition der *Continuous Stochastic Logic*(CSL) sowie den für das Model Checking der Kläranlage notwendigen Komponenten. Die hier verwendeten Definitionen und Bezeichnungen entstammen, wie bereits die des Abschnitt 5.1, [12]

Definition 11. Die Syntax einer CSL Formel ist analog zu [12] wie folgt definiert:

$$\begin{aligned} \Phi &::= \mathit{true} \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathbf{P}_{\sim p}[\phi] \mid \mathbf{S}_{\sim p}[\phi]; \\ \phi &::= \mathbf{X}\,\Phi \mid \Phi_1\ \mathbf{U}^I\ \Phi_2. \end{aligned}$$

Hierbei bezeichnet a eine atomare Eigenschaft, $\sim \in \{<, \leq, \geq, >\}$, $p \in [0,1]$ und I ein Intervall in $\mathbb{R}_{\geq 0}$.

In der oben beschriebenen Syntax wird zwischen zwei Typen von Formeln unterschieden. Zum einen den *Zustandsformeln* Φ, die sich auf jeweils einen der Zustände der CTMC beziehen, und den *Pfadformeln* ϕ, welche sich auf Pfade die in der CTMC genommen werden können beziehen. Zur Spezifikation einer Eigenschaft der CTMC wird stets eine Zustandsformel verwendet, Pfadformeln werden lediglich als Parameter der Operatoren $\mathbf{P}_{\sim p}[\phi]$ und $\mathbf{S}_{\sim p}[\phi]$. Ein Zustand s der CTMC $\mathcal{C}$ erfüllt die Formel $\mathbf{P}_{\sim p}[\phi]$ genau dann, wenn die Wahrscheinlichkeit den Pfad ϕ von s aus zu wählen im Intervall $\sim p$ liegt.

Es werden zwei Pfadformeln definiert zum einen der $\mathbf{X}$ ("next") und der $\mathbf{U}^I$ ("time bounded until") Operator. Der next-Operator beschreibt, dass der direkt folgende Zustand die Formel Ψ erfüllt. Der until-Operator hingegen besagt, das die Formel Φ_1 solange zutrifft, bis die Formel Φ_2 gilt. Bei einem "time-bounded until"muss die Formel Φ_2 in einem Zeitraum erfüllt sein der durch das Intervall I bestimmt ist. Überschreitet die Formel diese Bedingung, ist andernfalls aber gültig, so ist sie dennoch nicht erfüllt. Auch bei einem "time-bounded until"muss die vorherige Eigenschaft Φ_1 im gesamten Intervall I gültig sein, bis die Eigenschaft Φ_2 erfüllt ist. Ein ünbounded until", also eine Until-Formel, die keiner zeitlichen Beschränkung unterliegt, lässt sich durch die Wahl der Intervalls I als $I = [0, \infty)$ ableiten.

Der S-Operator beschreibt das steady-state verhalten der CTMC. Somit ist die Formel $\mathbf{S}_{\sim p}[\phi]$ genau dann erfüllt, wenn die steady-state Wahrscheinlichkeit der Zustände die Φ erfüllen im Intervall $\sim p$ liegt. Mit der Bezeichnung $s \models \Phi$ wird ausgedrückt, dass die CSL Formel Φ im Zustand s erfüllt ist. Weiterhin wird mit $Sat(\Phi)$ die Menge der Zustände $\{s \in S \mid s \models \Phi\}$ bezeichnet die eine bestimmte Zustandsformel Φ erfüllen. Ebenso wird für einen Pfad ω mit $\omega \models \phi$ beschrieben, dass die Pfadformel ϕ auf dem Pfad ω erfüllt ist. Somit ist die CSL Semantik über CTMCs wie folgt definiert.

Definition 12. Sei $\mathcal{C} = (S, \overleftarrow{s}, \mathbf{R}, L)$ eine *labelled CTMC*, und $s \in S$ ein beliebiger Zustand, dann ist die Relation $s \models \Phi$ induktiv wie in [12] wie folgt definiert

$$\begin{aligned}
s \models \texttt{true} \quad & \text{für alle} \quad s \in S \\
s \models a \quad & \Leftrightarrow \quad a \in L(s) \\
s \models \neg\Phi \quad & \Leftrightarrow \quad s \not\models \Phi \\
s \models \Phi \wedge \Psi \quad & \Leftrightarrow \quad s \models \Phi \wedge s \models \Psi \\
s \models P_{\sim p}[\phi] \quad & \Leftrightarrow \quad Prob^{\mathcal{C}}(s, \phi) \sim p \\
s \models S_{\sim p}[\Phi] \quad & \Leftrightarrow \quad \sum_{s' \models \Phi} \pi^{\mathcal{C}}_s(s') \sim p,
\end{aligned}$$

mit

$$Prob^{\mathcal{C}}(s, \phi) \stackrel{\text{def}}{=} Pr_s\{\omega \in Path^{\mathcal{C}}(s) \mid \omega \models \phi\},$$

und für einen beliebigen Pfad $\omega \in Path^{\mathcal{C}}(s)$:

$$\begin{aligned}
\omega \models \mathbf{X}\Phi \quad & \Leftrightarrow \quad \omega(1) \text{ ist definiert und } \omega(1) \models \Phi, \\
\omega \models \Phi \, \mathbf{U}^I \, \Psi \quad & \Leftrightarrow \quad \exists t \in I.(\omega@t \models \Psi \wedge \forall x \in [0, t).(w@x \models \Phi)).
\end{aligned}$$

Weiterhin können für Pfadformeln die beiden Operatoren $\Diamond$ *eventually* und $\Box$ *always* verwendet werden. $\Diamond\Phi$ *eventually* beschreibt, dass die Formel Φ irgendwann auf dem entsprechenden Pfad erfüllt wird. Für den Operator $\Box$ *always* gilt, dass auf einem Pfad, der $\Box\Phi$ erfüllt jeder der Zustände des Pfades die Eigenschaft Φ erfüllt.

Die beiden Operatoren $\Diamond$ und $\Box$ lassen sich wie folgt auf die bisherigen Definitionen zurückführen:

$$
\begin{aligned}
P_{\sim p}[\Diamond^I \Phi] &\equiv P_{\sim p}[\texttt{true}\ \mathbf{U}^I\ \Phi], \\
P_{\sim p}[\Box^I \Phi] &\equiv P_{\sim 1-p}[\Diamond^I \neg \Phi].
\end{aligned}
$$

Beispiel 5. Im Folgenden ein paar Beispiele für mögliche CSL Formeln:

- $S_{>0.99}[\mathit{stable}]$ Die Wahrscheinlichkeit, dass das System auf lange Sicht hin in einem stabilen Zustand befindet, ist größer als 0.99.
- $P_{<0.02}[\mathbf{X}\ \mathit{fail}]$ Die Wahrscheinlichkeit im nächsten Schritt in einen fehlerhaften Zustand zu gelangen ist kleiner als 0.02.
- $P_{>0.7}[\mathit{recovery}\ \mathbf{U}^{[0,3]}\ \mathit{stable}]$ Das Die Wahrscheinlichkeit, dass sich das System zwischen 0 und 3 Tagen im Wiederherstellungsmodus befindet, ohne das weitere Fehler auftreten, bevor das System wieder in einen stabilen Zustand gelangt, ist größer als 0.7.

6 Die Model Checker PRISM und MRMC

Dieses Kapitel befasst sich mit der Funktionsweise und dem Aufbau der Model Checking Tools PRISM [14] und MRMC [20]. Hierbei wird ein kurzer Überblick über die unterstützten Modelltypen, die Funktionsweise der Tools und die Erstellung von Modellen mithilfe der Tools gegeben.

6.1 PRISM

Der Probabilistic Symbolic Model Checker (PRISM) ist ein Tool, welches in Großbritannien an der Universität von Oxford entwickelt wird. Zum aktuellen Zeitpunkt unterstützt PRISM die Modellierung von DTMCs, CTMCs, MDPs, PAs und PTAs.

Zur Erstellung der Modelle wird die von PRISM eigene Sprache verwendet. In Listing 6.1 ist eine Modelldefinition in der PRISM Sprache am Beispiel einer Pumpe zu sehen, welche bereits als Beispiel in Abbildung 5.1 verwendet wurde. In Zeile 1 wird zunächst der Modelltyp festgelegt. In unserem Fall handelt es sich um ein *ctmc* Modell. In Zeile 3 bis 5 sind Deklarationen von Konstanten zu sehen. Diese sind optional und wurden lediglich für eine bessere Übersicht dem Modell vorangestellt. Es werden drei Raten definiert, welche den drei Transitionen der Pumpe entsprechen. An dieser Stelle dienen die vorherigen Definitionen der Konstanten lediglich der Übersicht, da die jeweiligen Raten nur jeweils ein Mal im Modell auftreten.

Bei größeren Modellen mit vielen Transitionen die jedoch einheitliche Raten besitzen ist die Definition von Konstanten sehr sinnvoll, da bei einer Änderung der Rate lediglich ein Wert geändert werden muss und der Rest des Modells unangetastet bleiben kann. In Zeile 3 bis 5 sowie 7 sind zudem Kommentare zu sehen die wie in vielen Sprachen

üblich durch einen vorangestelltes // eingeführt werden. In Zeile 8 beginnt dann die Definition der einzelnen Module des Modells. Diese können auf zwei verschiedene Weisen definiert werden. Zum einen explizit wie im Fall der ersten Pumpe in Zeile 8 – 15 zum anderen ist es möglich eine Moduldefinition zu kopieren wie es in Zeile 17 für die zweite Pumpe zu sehen ist.

```
ctmc

const double pu_dr = 1/180; //   6 Monate Damage Rate
const double pu_rr = 12; // 2h  Repair Rate
const double pu_ddr = 2; // 12h Damage Detection Rate

// Pumpe
module pump
    state_pu: [0..2] init 0;

    [] state_pu=0 -> pu_dr  : (state_pu'=1);
    [] state_pu=1 -> pu_ddr : (state_pu'=2);
    [] state_pu=2 -> pu_rr  : (state_pu'=0);

endmodule

module pump2 = pump[state_pu = state_pu2] endmodule

label "Stable" = state_pu = 0 & state_pu2=0;
```

Listing 6.1: Definition zweier Pumpe mithilfe der PRISM Language.

Ein explizit beschriebenes Modul folgt immer einer festen Syntax. Zunächst wird einer oder mehrere Zustandsräume definiert, welcher die Zustände der beschriebenen CTMC widerspiegelt, wie in Zeile 9 zu sehen ist. Hierbei wird zunächst ein Name des Zustandsraums festgelegt in diesem Fall `state_pu` für den Zustandsraum der Pumpe. Im Folgenden wird ein Intervall angegeben, welches die Größe des Zustandsraums angibt, in diesem Fall [0..2], da nur diskrete Zustandswerte definiert werden können, beschreibt dies somit die drei Zustände 0, 1 und 2 welche den Zuständen s_0, s_1 und s_2 entsprechen,

welche in Abbildung 5.1 zu sehen sind. Neben der Definition des Zustandsraumes muss noch ein Startzustand bestimmt werden. Dieser wird mit der Notation `init0` in diesem Beispiel dem Zustand 0 zugewiesen.

Nach der Definition der Zustände der CTMC müssen noch die einzelnen Transitionen definiert werden, diese sind in den Zeilen 11 bis 13 zu sehen. Die Definition einer oder mehrerer Transitionen wird zunächst immer mit den Klammern [] eingeleitet. Diese können *Label* enthalten, welche die entsprechenden Transitionen beschreibt. Zudem dient diese Art der Beschriftung der Synchronisation verschiedener Transitionen. Soll eine Transition immer genau dann schalten, wenn es eine andere tut, so ist es möglich beide Transitionen mit dem gleichen Label zu versehen, um dies zu gewährleisten. Neben der Beschriftung der Transitionen muss noch der Ausgangs- sowie Endzustand der jeweiligen Transitionen festgelegt werden. Der Ausgangszustand der Transitionen wird stets zuerst bestimmt z.B. in Zeile 11 `state_pu` $= 0$, daraufhin folgt immer ein Pfeil $->$. Im Folgenden können nun Folgezustände mit der jeweiligen Rate angegeben werden in der Syntax `<rate>` : (`<zustandsraum>`$'$ $=$ `<zustandsnummer>`). In Zeile 11 ist ein Übergang mit der Rate `pu_dr`, welche zuvor als Konstante mit dem Wert $\frac{1}{180}$ festgelegt wurde, von Zustand 0 zu Zustand 1 zu sehen. Möchte man mehrere Zustände als Folgezustand angeben, also mehrere Transitionen mit dem selben Startzustand zugleich definieren, so können die Raten und Zustandsbestimmungen mit einem $+$ getrennt hintereinander folgen. Die Definition der Transitionen wird immer mit einem ; beendet.

Wie bereits erwähnt ist es möglich, Module auch als Kopie eines bestehenden Moduls zu definieren. Dies ist in Zeile 17 zu sehen. Die Definition beginnt und endet wie bei einer regulären Definition mit `module` und `endmodule`. Zudem muss lediglich der Name des neuen Modells `pump2`, sowie ein neuer Zustandsraum `[state_pu = state_pu2]`, angegeben werden, um eine Kopie des bestehenden Moduls zu erstellen. Wird ein CTMC Modell mithilfe mehrerer Module definiert, so wie es hier der Fall ist, wird eine Parallele Komposition der einzelnen Module gebildet. Diese werden ohne spezielle Angabe weiterer Informationen lediglich an gleichen Labeln synchronisiert. Zudem sind andere Formen der Synchronisation möglich welche hier definiert sind [2].

Neben der Vergabe von Labeln an Transitionen ist es auch möglich, Zustände mit Labeln zu versehen. Wie in Zeile 19 zu sehen ist, befindet sich die CTMC im Zustand "Stable"genau dann, wenn sowohl Pumpe 1 als auch Pumpe 2 sich im Zustand 0 befinden. Diese Form von Labeln vereinfacht zudem das spätere Model Checking, da die Label als spätere Zustandsformeln verwendet werden können.

```
P=? [ true U<=20 "Stable" ]

S>0.6 [ state_pu = 0 ]
```

Listing 6.2: Beispiele für CSL Formeln in der PRISM Language.

Neben der `.pm` Datei zur Definition des Modells verwendet PRISM noch `.props` Dateien, welche die Formeln enthalten, welche für das Model Checking benötigt werden. Eine solche Definition ist Beispielhaft in Listing 6.2 zu sehen. Wie zu erkennen ist, unterscheidet sich die Syntax kaum von der in Abschnitt 5.3 beschriebenen CSL Formeln. Wie bereits erwähnt ist es möglich, die zuvor definierten Zustandslabel innerhalb der Formeln zu verwenden wie es in Zeile 1 zu sehen ist. Zudem kann für das Intervall der Wahrscheinlichkeit bei den einzelnen Operatoren zudem der Ausdruck $=?$ verwendet werden, welcher die genaue Wahrscheinlichkeit der geprüften Formel bestimmt.

Des Weiteren verfügt PRISM über eine vollständige GUI welche in Abbildung 6.1 zu sehen ist. Diese umfasst einen Code Editor zur Erstellung der bereits beschriebenen Modelle, sowie einen Formeleditor zur Spezifikation von Formeln in allen unterstützten Logiken. Zudem verfügt die GUI über die Möglichkeit Plots zu erstellen, die z.B. die Ergebnisse einer CSL Formel unter Veränderung einer Variablen beschreiben. Darüber hinaus verfügt die PRISM GUI über einen Simulator, mit dem der Betrieb eines Modells simuliert werden kann und der Nutzer manuell Pfade innerhalb des Modells auswählen kann, um das Verhalten des Modells zu untersuchen.

Die Nutzeroberfläche von PRISM wurde in Java entwickelt, wohingegen die meisten Model Checking Algorithmen in C++ implementiert wurden. Zur internen Repräsentation der Zustandsräume kann durch den Nutzer bestimmt werden, mögliche Optionen sind *ßparse matrices"*, *MTBDDs* sowie eine *hybrid engine* welche beide Ansätze kombiniert. Die einzelnen Repräsentationen unterscheiden sich unter anderem in der Größe der berechenbaren Modelle, der Berechnungsgeschwindigkeit sowie des benötigten Speicherplatzes. Weiterhin verfügt PRISM über eine Auswahl vieler gängiger

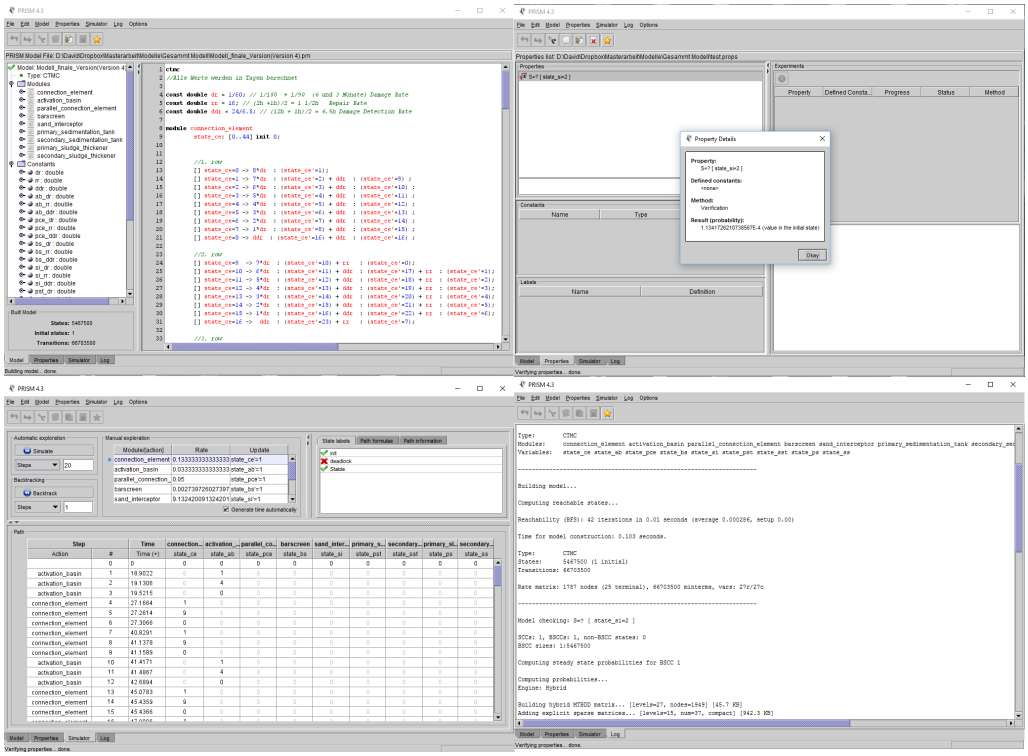

Abbildung 6.1: Beispiel der PRISM GUI.

Methoden zur Berechnung von Transienten Wahrscheinlichkeiten, das Lösen linearer Gleichungen sowie das Lösen von Formeln auf MDPs. Des Weiteren lässt sich die Genauigkeit der Rechenmethoden anhand von Epsilon Genauigkeit, Abbruchkriterien sowie der Anzahl durchgeführter Iterationen festlegen. Außerdem verfügt PRISM über die Möglichkeit Modell, die in der PRISM Sprache definiert wurden in verschiedenen Weisen zu exportieren, unter anderem auch in einem für den Model Checker MRMC lesbaren Format. All dies macht PRISM zu einem hervorragenden Tool für viele Anwendungen des Quantitativen Modell Checkings.

6.2 MRMC

Der *Markov Reward Model Checker (MRMC)* [20] ist wie auch PRISM in der Lage DTMCs und CTMCs als Modelle zu verarbeiten. Da MRMC sich jedoch auf das Checken von Modellen mit *Rewards* spezialisiert werden noch verschiedene andere Modelle wie DMRMs, CMRMs und CTMDPIs unterstützt. Für die entsprechenden Modell Typen werden auch die korrespondieren Logiken PCTL, CSL sowie PRCTL und CSRL

unterstützt. Die Beschreibung der in MRMC erstellten Modelle erfolgt über mehrere Dateien. Eine `.tra` Datei, in der der Aufbau des Modells in Form der Raten- oder Wahrscheinlichkeitsmatrix beschrieben ist. Eine `.lab` Datei die die zu den einzelnen Zuständen gehörenden Labels enthält sowie eine `.rew` Datei welche die Rewards der einzelnen Zustände speichert.

Pumpe.tra	Pumpe.lab	Pumpe.rew

`STATES 3`
`TRANSITIONS 3`
 	
`0 1 0.00555555555`
`1 2 2.0`
`2 1 12.0`
	`#DECLARATION`
	`Normalbetrieb`
	
	`Ausgefallen`
	`Reparieren`
	`#END`
	
	`1 Normalbetrieb`
	`2 Ausgefallen`
	`3 Reparieren`
		`1 5`
		`3 1`

Tabelle 6.1: Beispiel der einzelnen MRMC Eingabedateien anhand der bereits in Abbildung 5.1 verwendeten Pumpe.

In Tabelle 6.1 sind die einzelnen Eingabedateien für die in Abbildung 5.1 bereits als Beispiel verwendete Pumpe zu sehen. In der `.tra` Datei werden zunächst im Header der Datei die Anzahl der Zustände und Transitionen definiert. Im Folgenden wird dann jeweils eine Transition pro Zeile aufgelistet. Dies erfolgt in der Form `<Ausgangszustand> <Folgezustand> <Rate>`. Auf diese sehr simple Weise lässt sich die Ratenmatrix einer CTMC abbilden, jedoch ist die Beschreibung großer Modelle mit entsprechendem Schreibaufwand verbunden. Diesbezüglich verweisen die Entwickler auf den Export von in PRISM erstellten Modellen [20].

Die `.lab` Datei speichert die einzelnen Zustandslabel des Modells. In dem hier verwendeten Beispiel existieren die Label `Normalbetrieb`, `Ausgefallen` und `Reparieren`. Alle verwendeten Label müssen zunächst im Header der Datei deklariert werden. Daraufhin können sie dann den einzelnen Zuständen zugewiesen werden. Dies erfolgt in der Form `<Zustand> <Liste der Label>`. In dem hier gezeigten Beispiel wird jedem Zustand nur jeweils ein Label zugewiesen, es ist jedoch auch möglich mehrere Label pro Zustand zu vergeben.

Die `.rew` Datei speichert die im Modell vergebenen Rewards. Da unser Beispiel keine Rewards verwendet wurden hier beispielhaft dem Zustand 1 ein Reward von 5 und dem Zustand 3 ein Reward von 1 gegeben. Die Datei besitzt keinen Header, die einzelnen Rewards werden in der Form `<Zustand> <Reward>` vergeben.

```
P>0.5 [ tt U[0,20] Normalbetrieb ]
```

Listing 6.3: Definition einer CSL Formel mit MRMC.

Das Model Checking erfolgt bei MRMC über einen Programm Prompt, in dem die einzelnen Formeln der Logik eingegeben werden können. Die Syntax unterscheidet sich nur gering von der in PRISM verwendeten Syntax. Listing 6.3 zeigt ein time-bounded until als CSL Formel. Hierbei steht `tt` für `true` der bound des until-Operators wird als Intervall `[0,20]` angegeben. MRMC ist zudem in der Lage Bisimulation Minimization auf die verwendeten Modelltypen anzuwenden, um die Laufzeit des Model Checkings in vielen Fällen stark zu verringern. Des Weiteren besitzt MRMC eine Erweiterung der Sprachen zum checken von verschiedenen Rewards, diese werden jedoch in dieser Arbeit nicht verwendet daher wird an dieser Stelle auf [20] verwiesen.

7 Modellierung

Dieses Kapitel befasst sich mit der Modellierung der Kläranlage als späteres Modell, welches die Grundlage des Model Checkings darstellt. Hierzu werden zunächst die grundlegenden Überlegungen dargelegt in welcher Form die Kläranlage modelliert werden soll. Daraufhin wird der genaue Aufbau der in dieser Arbeit betrachteten Kläranlage beschrieben Abschnitt 7.2. Der nächste Abschnitt befasst sich dann mit den Ergebnissen einer FMEA Analyse und wie diese verwendet werden können Abschnitt 7.3. In Abschnitt 7.4 beschrieben wie die Ergebnisse der FMEA Analyse bei der Erstellung des Modells verwendet werden können. Das Ende dieses Kapitels bildet Abschnitt 7.5 in dem erläutert wird wie die separat modellierten Komponenten der Kläranlage zu einem vollständigen Modell der Anlage zusammengefasst werden können.

7.1 Konzept zur Modellierung der Kläranlage

Wie in Kapitel 3 bereits zu erkennen ist, lässt sich der Aufbau einer Kläranlage auf viele verschiedene Weisen betrachten und entsprechend dieser Gesichtspunkte modellieren. In dieser Arbeit wurde eine Modellierung gewählt, bei der das Verhalten der Kläranlage über einen beliebigen Zeitraum hinweg betrachtet werden kann. Hierbei wurden CTMC als Modelltyp gewählt, da diese es erlauben beliebige Raten für die Übergänge zwischen den einzelnen Zuständen zu wählen. Die einzelnen Zustände der CTMC bilden ab, in welchem Zustand sich die Kläranlage gerade befindet. Hierbei gibt es für jede Komponente die modelliert wird verschiedene Zustände wie z. B. ”Normalbetrieb” oder ”Beschädigt” in der sie sich befinden kann. Im Gesamtmodell der Kläranlage spiegelt dann ein Zustand jeweils den Zustand aller Teilkomponenten wieder z. B. ”alle Komponenten befinden sich im Normalbetrieb” oder ”Das Vorklärbecken und der Sandfang sind beschädigt, alle anderen Komponenten sind im Normalbetrieb”. Die

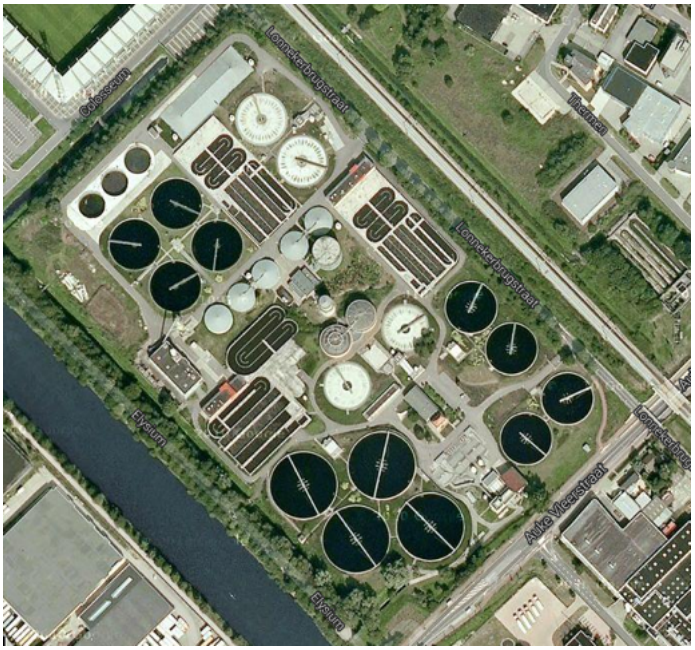

Abbildung 7.1: Kläranlage in Enschede aus der Vogelperspektive.[6]

Übergänge der CTMC sind mit entsprechenden Raten versehen. Hierbei entspricht eine Rate von 1 einem Tag. Wie für eine CTMC üblich ist die Wahl welche Transition schaltet, falls mehrere Möglichkeiten bestehen, eine *race condition*. Dies stellt eine leichte Abstraktion zur Realität dar, da theoretisch in der genau gleichen Sekunde mehrere Komponenten der Kläranlage beschädigt werden könnten. Allerdings stellt es keinen Unterschied für den Betrieb einer Kläranlage dar ob eine Komponente zur exakt selben Zeit wie eine andere ein Fehlversagen aufweist oder wenige Sekunden später. Dies sollte einen groben Überblick über das hier entwickelte Modell geben. Wie dies genau für die hier betrachtete Kläranlage umgesetzt ist wird in Abschnitt 5.2 sowie Abschnitt 7.5 beschrieben.

7.2 Aufbau der Kläranlage

Nachdem bereits in Kapitel 2 der grundlegende Aufbau einer Kläranlage, sowie die Funktionsweise der einzelnen Komponenten beschrieben wurde, wird nun ein konkreter Aufbau benötigt, um das Modell zu erstellen. Als Grundlage für das hier verwendete Modell dient die Kläranlage der Stadt Enschede in den Niederlanden, die in [6] bereits als Ausgangspunkt einer Untersuchung diente. Abbildung 7.1 Zeigt die

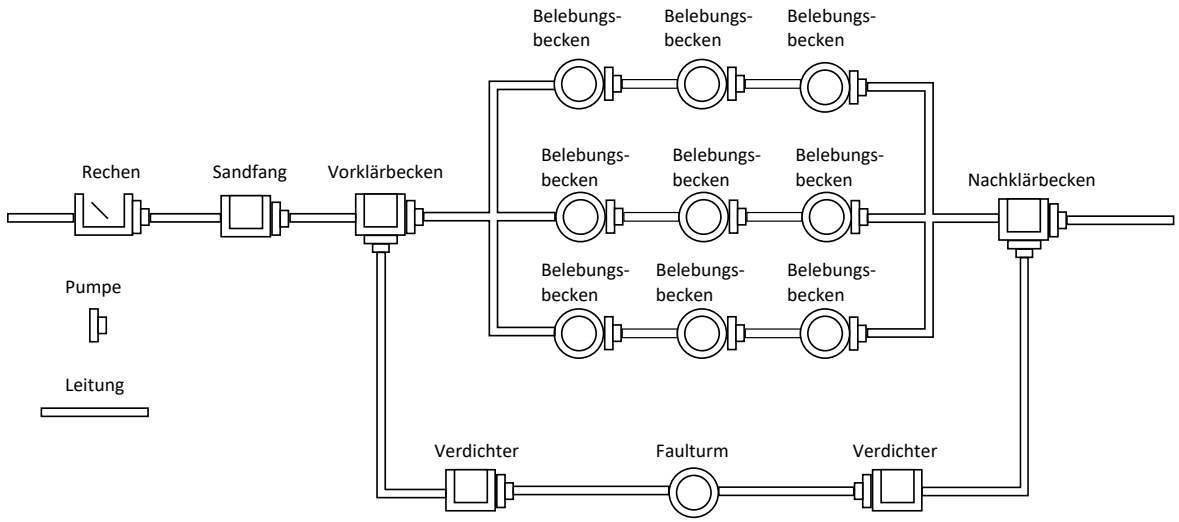

Abbildung 7.2: Schematische Darstellung der Kläranlage.

Kläranlage aus der Vogelperspektive. Hierbei sind die einzelnen Klärbecken gut zu erkennen. Ein schematischer Aufbau der Anlage ist in Abbildung 7.2 zu sehen. Das Schema zeigt den genauen Aufbau der Anlage. Diese unterscheidet sich zu den vorherigen Beschreibungen lediglich durch die drei Reinigungsstraßen, die aus jeweils 3 Belebungsbecken bestehen, von dem zuvor beschriebenen Aufbau einer Kläranlage.

Wie bereits in Kapitel 2 beschrieben, durchläuft das Schmutzwasser zunächst den Rechen und Sandfang, bevor es in das Vorklärbecken geleitet wird, wo sich Sand und grobe Verunreinigungen absetzen können. Somit verläuft die mechanische Reinigung genau so, wie in Kapitel 2 beschrieben. Die biologische Reinigung in dieser Kläranlage erfolgt jedoch in drei Reinigungsstraßen, die jeweils aus drei hintereinander geschalteten Belebungsbecken bestehen, bevor das Schmutzwasser wieder in einem gemeinsamen Nachklärbecken zusammenfließt. Zudem gibt es zwei getrennte Verdichter, je einen für das Vor- und einen für das Nachklärbecken, die dem abgetragenen Schlamm das restliche Wasser entziehen.

Tabelle 7.1: FMEA Tabelle

Komponente	Severity	Occurrence	Detection	Rate
Rechen	7	4	6	168
Sandfang	7	3	6	126
Vorklärbecken	4	4	5	80
Belebungsbecken	6	6	3	108
Nachklärbecken Schlammbildung	6	7	3	126
Nachklärbecken Beschädigung	5	4	4	80
Verdichter	4	2	7	56
Pumpe	4	5	4	80
Leitung	3	6	2	36

7.3 FMEA Analyse der Kläranlage

Das spätere Modell, dass hier entwickelt wird, basiert auf den Ergebnissen einer FMEA Analyse der vorliegenden Kläranlage. Hierbei werden die für die einzelnen Komponenten vergebenen Werte in Raten umgewandelt, welche dann im entwickelten CTMC Modell verwendet werden können. Wie in Kapitel 3 zu sehen existieren bereits andere Arbeiten, die auf verschiedenen Betrachtungsebenen FMEA Analysen für ähnliche Kläranlagen durchführen. Da die Durchführung einer eigenen Analyse für diese Arbeit dessen Rahmen gesprengt hätte und es nicht deren Fokus darstellt, wurde auf Schätzwerte zurückgegriffen. Die so getroffenen Annahmen basieren auf den Ergebnissen folgender Arbeiten [7] und haben lediglich Auswirkungen auf die später erhaltenen Ergebnisse des Model Checkings jedoch nicht auf den Modellierungsprozess. Somit ist es möglich für spätere Arbeiten möglich lediglich die Ergebnisse der FMEA Analyse anzupassen solange der Aufbau der Kläranlage identisch ist.

Wir erhalten somit die Ergebnisse, die in Tabelle 7.1 zu sehen sind. Für jede der zuvor beschriebenen Komponenten der Anlage liegen nun drei Kenngrößen vor. Die *Severity*, die beschreibt, wie schwerwiegend die Folgen des Auftretens eines Fehlers in dieser Komponente sind. *Occurrence* beschreibt die Häufigkeit, mit der die entsprechende Komponente versagt, sowie die *Detection* Rate die angibt, wie wahrscheinlich es ist, dass ein eintretender Fehler auch bemerkt wird.

Nun müssen diese Angaben in ein entsprechendes Modell der Anlage umgewandelt werden. Die *Severity* also die Schwere des auftretenden Fehlers spielt hierbei zunächst eine untergeordnete Rolle, da sie nicht mit modelliert wird.

Tabelle 7.2: Detektionsdauer pro Rate

Rating	Definition	Time
10	Der Fehler wird oder kann nicht erkannt werden	mehr als 1 Jahr
9	Sehr entfernte Chance das der Fehler erkannt wird	1 Jahr
8	Entfernte Chance das der Fehler erkannt wird	6 Monate
7	Sehr geringe Chance das der Fehler erkannt wird	1 Monat
6	Geringe Chance das der Fehler erkannt wird	1 Woche
5	Moderate Chance das der Fehler erkannt wird	1 Tag
4	Mäßig hohe Chance das der Fehler erkannt wird	12 Stunden
3	Hohe Chance das der Fehler erkannt wird	6 Stunden
2	Sehr hohe Chance das der Fehler erkannt wird	1 Stunde
1	Der Fehler wird fast sicher erkannt	wenige Minuten

Für die Erstellung des Modells wird zunächst jede Komponente isoliert betrachtet. Für jeden der möglichen Fehler, die auftreten können, wird eine Auftrittswahrscheinlichkeit, eine Wahrscheinlichkeit, mit der der Fehler erkannt wird und eine Reparaturdauer benötigt. Die Auftrittswahrscheinlichkeit kann in direkter Form aus der FMEA entnommen werden. Jedem der Ratings steht wie in Tabelle 4.2 zu sehen ist eine Zeitspanne gegenüber, in der die entsprechende Komponente voraussichtlich versagen oder einen Schaden erleiden wird. Diese Zeitangabe lässt sich direkt als eine der Transitionen der späteren CTMC abbilden.

Die Erkennungsrate, mit der ein Fehler detektiert wird, lässt sich hingegen nicht direkt aus der FMEA entnehmen. Wie Tabelle 4.3 zeigt, gibt es für jedes der Ratings nur eine grobe textuelle Beschreibung für das Auffinden des Fehlers. In manchen Varianten der FMEA findet sich an dieser Stelle auch eine ungefähre Wahrscheinlichkeit, mit der der Fehler erkannt wird. Auch dies ist für die Modellierung der CTMC nicht ausreichend, da wir einen Zeitraum benötigen, in dem der vorliegende Fehler erkannt wird. Aus diesem Grund wurde anhand der Detektionsrate versucht, wie bereits bei der Auftrittswahrscheinlichkeit ein Zeitraum zu finden, in dem ein Fehler mit der entsprechenden Detektionsrate sicher erkannt wird. Das Ergebnis ist in Tabelle 7.2 zu sehen.

Die hier getroffene Annahme ist, dass ein beliebiger Fehler innerhalb eines gewissen Zeitraums immer erkannt wird. Wird bei einer standardmäßigen Inspektion ein Fehler nur mit einer gewissen Wahrscheinlichkeit gefunden, so kann man davon ausgehen, dass nach einer ausreichenden Anzahl von durchgeführten Inspektionen der Fehler irgendwann gefunden wird. Zudem besteht die Möglichkeit, dass Fehler bei täglichen

Tabelle 7.3: Reparatur Zeiten

Komponente	Reparaturdauer
Rechen	4h
Sandfang	3h
Vorklärbecken	4h
Belebungsbecken	10h
Nachklärbecken	10h
Verdichter	3h
Pumpe	2h
Leitung	1h

Routineinspektionen nicht erkannt werden können jedoch bei einer gründlichen Jahresinspektion klar zu erkennen sind. Auf Grundlage dieser Annahme ist es nun möglich jeder Detektionsrate einen Zeitraum zuzuordnen, in dem der entsprechende Fehler als solcher erkannt wird.

Die letzte Größe, die für die Erstellung der CTMC fehlt, ist eine Reparaturrate, also eine Zeitangabe, wie lange es dauert, bis eine beschädigte Komponente ausgetauscht oder repariert ist und das System wieder mit voller Leistung weiterarbeiten kann. Diese Rate lässt sich jedoch nicht anhand der FMEA Analyse herleiten. Es besteht zwar in einigen Fällen eine Korrelation zwischen der Reparaturdauer einer Komponente und der Schwere eines aufgetretenen Fehlers (*Severity*). Jedoch wird dieser Wert auch noch durch viele andere Faktoren bestimmt wie z.B. die Kosten der Reparatur, mögliche Personenschäden, die Verletzung von gesetzlichen Vorgaben oder der Verlust von Kunden. Jedoch kann das *Severity* Rating beim späteren Model Checking genutzt werden, um besonders kritische Komponenten zu identifizieren, die eine genauere Betrachtung benötigen. Da die eigentliche FMEA keinen Ausschluss darüber gibt, wie lange die Reparatur der einzelnen Komponenten benötigt wurde, auch hier auf Schätzwerte zurückgegriffen, welche in Tabelle 7.3 zu sehen sind. In der Praxis sollte das Expertenteam, dass für die Erstellung der FMEA Analyse verantwortlich ist in der Lage sein, akkurate Werte für die Reparaturdauer, entweder durch eine genaue Analyse der verbauten Komponenten oder durch das zurückgreifen auf Erfahrungswerte, zu liefern. Mit den so erhaltenen Werten ist es nun möglich, die Kläranlage als CTMC zu modellieren.

7.4 Modellierung der Kläranlage als CTMC

Nachdem in Abschnitt 7.3 nun die Ergebnisse der FMEA Analyse so weit aufbereitet wurden, dass sie zur Erstellung eines CTMC Modells verwendet werden können befasst sich dieser Abschnitt nun mit der Erstellung dieses Modells sowie dessen Umsetzung in PRISM.

Abbildung 7.3 zeigt die Modellierung einer einzelnen Komponente sowohl in allgemeiner Form als auch am Beispiel des Rechens. Das Teilmodell besteht aus drei Zuständen und drei Transitionen, die die jeweiligen Zustände der Komponente widerspiegeln. Im Ausgangszustand s_0 ist die Komponente voll funktionsfähig. Die Rate λ entspricht der Fehlerauftrittswahrscheinlichkeit (*Ocurrence*) mit dem die Komponente in den Zustand s_1 übergehen kann. In diesem Zustand ist die Komponente beschädigt und bereits nicht mehr einsatzfähig oder arbeitet mit verminderter Leistung. Mit der Detektionsrate μ kann der eingetretene Fehler dann erkannt werden und in den Zustand s_2 übergehen in dem er immer noch funktionsuntüchtig ist, jedoch nun repariert wird. Über die Reparaturdauer ρ gelangt die Komponente dann in ihren Ausgangszustand s_0 zurück. Die Modellierung der Komponenten in PRISM erfolgt wie in Listing 6.1 beschrieben. Der zu dem in Abbildung 7.3 beschriebenen Rechen ist in Listing 7.1 zu sehen

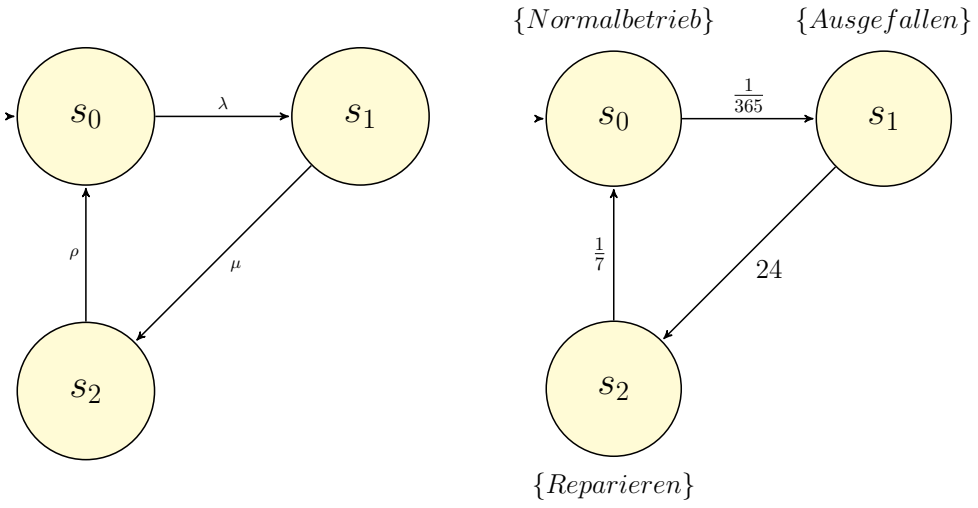

Abbildung 7.3: Beispielmodellierung einer beliebigen Komponente der Kläranlage mit einem Fehlertyp (links). Explizite Beschreibung des Rechens als CTMC (rechts).

```
// Rechen
const double bs_dr = 1/365; //1 Jahr Damage Rate
const double bs_rr = 24; //1h Repair Rate
const double bs_ddr = 1/7; //1 Woche Damage Detection Rate

module barscreen
    state_bs: [0..2] init 0;

    [] state_bs=0 -> bs_dr  : (state_bs'=1);
    [] state_bs=1 -> bs_ddr : (state_bs'=2);
    [] state_bs=2 -> bs_rr  : (state_bs'=0);
endmodule
```

Listing 7.1: Definition des Rechen in der PRISM Language.

Gibt es für die Komponente verschiedene Fehler, die auftreten können, so wie es für das Nachklärbecken der hier betrachteten Kläranlage der Fall ist, werden diese durch zusätzliche "Kreisläufe" von Zuständen abgebildet. Hierbei wird davon ausgegangen, dass die einzelnen Fehler unabhängig voneinander auftreten. Im Falle des Nachklärbeckens ist die sehr plausibel, da der Betrieb des Beckens bei dem Eintreten eines der beiden Fehler ausgesetzt werden muss bevor der Fehler behoben werden kann. Eine solche CTMC ist in Abbildung 7.4 zu sehen. Hier wird zum einen zwischen der Beschädigung des Beckens sowie der Bildung von Schwimmschlamm unterschieden. Die Bildung von Schwimmschlamm ist eines der größten Probleme, die bei der Reinigung von Abwasser mithilfe von Bakterienkulturen auftreten kann. Aus diesem Grund wurde dessen Auftreten separat von anderen Schäden und Fehlern mit modelliert. Dies ermöglicht es bei dem späteren Model Checking diese Fehlerquelle isoliert zu betrachten. Die Modellierung des Nachklärbeckens als CTMC mit PRISM ist zudem in Listing 7.2 beschrieben. Es wäre zudem möglich weitere ausschlaggebende Fehlerquellen mit zu modellieren, jedoch ist hierbei zu berücksichtigen, dass bereits geringfügige Änderungen zu einer enormen Vergrößerung des späteren Gesamtmodells führen können. Im Falle von Fehlern die sich gegenseitig beeinflussen ist es ebenso möglich dies in Form einer entsprechenden CTMC zu modellieren anstelle der hier verwendeten Modellierung unabhängiger Ereignisse.

```
// Nachklaerbecken
//Beschaedigung
const double sst_dr1 = 1/365; //1 Jahr Damage Rate
const double sst_rr1 = 4; //6h Repair Rate
const double sst_ddr1 = 2; //12h Damage Detection Rate
//Schwimmschlamm bildung
const double sst_dr2 = 1/30; //1 Monat Damage Rate
const double sst_rr2 = 2; //12h Repair Rate
const double sst_ddr2 = 4; //6h Damage Detection Rate

module secondary_sedimentation_tank
    state_sst: [0..4] init 0;

    [sst_damaged] state_sst=0 -> sst_dr1 : (state_sst'=1);
    [] state_sst=1 -> sst_ddr1 : (state_sst'=2);
    [] state_sst=2 -> sst_rr1 : (state_sst'=0);

    [sst_floating_sludge] state_sst=0 -> sst_dr2 : (
       state_sst'=3);
    [] state_sst=3 -> sst_ddr2 : (state_sst'=4);
    [] state_sst=4 -> sst_rr2 : (state_sst'=0);

endmodule
```

Listing 7.2: Definition des Nachklärbeckens in der PRISM Language.

7.5 Erstellung eines Gesamtmodells

Um nun aus den separat modellierten Komponenten der Kläranlage ein einheitliches Gesamtmodell zu bilden, auf dem das spätere Model Checking erfolgen kann, wird eine parallele Komposition der einzelnen Komponenten, wie in Abschnitt 5.1 beschrieben, vorgenommen. Wie im Schema von Abbildung 7.2 zu sehen ist, besteht die Kläranlage aus einer Vielzahl von Komponenten. Einem Rechen, Sandfang und Vorklärbecken, die seriell die für die mechanische Vorreinigung zuständig sind. Die chemische Reinigung

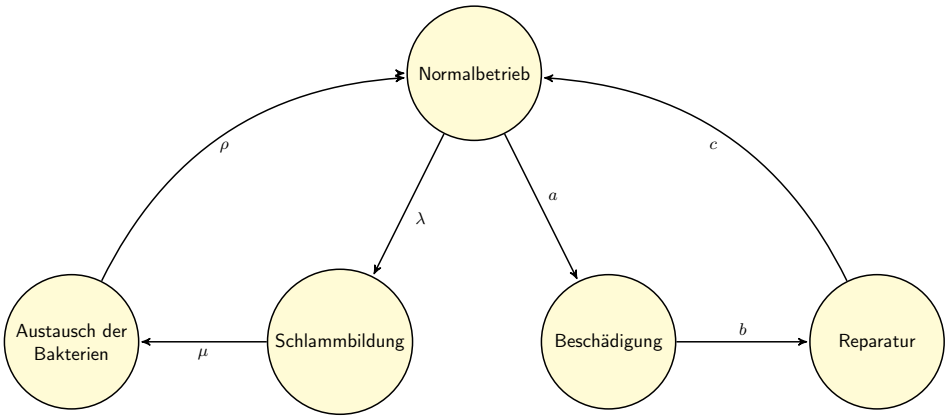

Abbildung 7.4: CTMC Modell des Nachklärbeckens mit zwei unabhängigen Fehlertypen.

erfolgt anfangs parallel auf drei Reinigungsstraßen mit je drei in Reihe geschalteten Belebungsbecken und fließt dann in ein gemeinsames Nachklärbecken, von dem aus das Wasser gereinigt in anliegende Gewässer entlassen wird. Vom Vor- und Nachklärbecken aus wird zudem der abgetragene Schlamm zu jeweils einem Verdichter geleitet, der den Wassergehalt der Masse reduziert. Der entwässerte Schlamm kann dann entsorgt oder gelagert werden. Alle Becken sind zudem mit Leitungen verbunden, die zudem je eine Pumpe für den Transport des Wassers besitzen. Somit enthält das Modell neben den einzelnen Becken der Kläranlage noch 16 Leitungen und 17 Pumpen.

Somit muss eine parallele Komposition von insgesamt 48 einzelnen CTMCs gebildet werden. Dies führt jedoch zu einem gigantischen Zustandsraum von mehr als $1.32 \cdot 10^{23}$ Zuständen und $6.4 \cdot 10^{24}$ Transitionen, vergleiche Listing 7.3. Eine solche Anzahl von Zuständen und Transitionen kann selbst mit modernen Model Checking Tools nicht bewältigt werden. Wie bereits in Kapitel 6 beschrieben ist PRISM in der Lage Model Checking auf Modellen mit bis zu $10^7 - 10^8$ Zuständen durchzuführen. Das bisherige Modell ist von dieser Größenordnung noch weit entfernt und muss daher minimiert werden.

```
Building model...

Computing reachable states...

Reachability (BFS): 98 iterations in 0.12 seconds (average ↵
    0.001173, setup 0.00)

Time for model construction: 0.241 seconds.

Type:        CTMC
States:      1.3294407179478 75E23 (1 initial)
Transitions: 6.407904260508759E24

Rate matrix: 2198 nodes (15 terminal), 6.407904260508759E24 ↵
     minterms, vars: 97r/97c
```

Listing 7.3: Ausgabe des PRISM Logs bei der Erstellung des Modells.

8 Minimierung des Modells

Wie in Kapitel 7 gezeigt wurde, ist es möglich, die Ergebnisse einer FMEA Analyse zu nutzen um ein Modell der Kläranlage, in Form einer CTMC zu erstellen. Jedoch hat sich hierbei gezeigt, dass eine naive Modellierung aller vorhandenen Komponenten und eine spätere Kopplung über eine parallele Komposition der einzelnen CTMCs zu einem Modell führt, das für weitere Betrachtungen zu groß ist. Aus diesem Grund muss das so erstellte Modell nun in seiner Größe minimiert werden, bis es eine Größe erreicht auf der bestehende Modell Checking Tools wie PRISM Berechnungen durchführen können.

8.1 Bisimulation

In Abschnitt 5.2 wurden bereits einige Verfahren zur Minimierung von CTMCs vorgestellt, welche in der Lage sind ein Modell so zu minimieren, dass kein oder nur ein geringer Verlust der Semantik auftritt. Eines dieser Verfahren ist die Minimierung durch Bisimulation. Wie in Kapitel 6 beschrieben unterstützt der PRISM Model Checker diese Funktionalität jedoch nicht. Aus diesem Grund sollte der MRMC Checker zur Verwendung dieser Methode genutzt werden. In Kapitel 6 wurde ebenfalls die Funktionsweise des MRMC Model Checkers erläutert. Dieser besitzt im Vergleich zu PRISM eine andere Art der Modellbeschreibung. Es ist entweder möglich die einzelnen Zustände manuell in der Definitionsdatei anzugeben wie in Tabelle 6.1 zu sehen ist. Darüber hinaus ist es möglich, eine solche Modelldatei als Exportdatei von PRISM erstellen zu lassen. Dies ermöglicht es, die komfortable Art der Modellbeschreibung von PRISM zu nutzen und zugleich die in einigen Bereichen umfangreicheren Minimierungs- und Model Checkingfunktionen des MRMC zu verwenden. Jedoch ist die Größe der Auslagerungsdatei hierbei entscheidend. Ein kurzes Experiment mit

einem Teilmodell mit lediglich 1594323 Zuständen und 20726199 Transitionen lieferte bereits eine Auslagerungsdatei von 344.2 MB. Ein Modell mit 4782969 Zuständen und 66961566 Transitionen erreicht bereits eine Größe von ca. 1.2 GB. Dies zeichnet ein lineares Wachstum der Dateigröße proportional zur Anzahl der Transitionen ab. Dies ist auch sehr plausibel, da die Datei pro Transition eine zusätzliche Zeile enthält. Gehen wir nun jedoch von unserem anfänglichen Modell mit $6.4 \cdot 10^{24}$ Transitionen aus würde dies eine Datei mit einer Größe von rund 106000 Zettabyte ergeben. Da dies eine weitere Verarbeitung unmöglich macht, wurde dieser Ansatz zur Modellminimierung verworfen.

8.2 Symmetrie Reduktion

Eine andere Art der Minimierung stellt die Symmetrie Reduktion dar, die ebenfalls in Abschnitt 5.2 beschrieben wurde. Für die Symmetrie Reduktion werden identische Teilmodelle von Komponenten benötigt, die dann über die Parallele Komposition zu einem großen Modell zusammengefasst werden. Dies ist bei der hier betrachteten Kläranlage für diverse Komponenten der Fall. So kann die Symmetrie Reduktion sowohl auf die diversen Pumpen und Leitungen als auch auf die Belebungsbecken der Anlage angewendet werden.

Jedoch besitzt die Symmetrie Reduktion gegenüber der Bisimulation einen Nachteil. Wendet man die Symmetrie Reduktion auf die Komposition der identischen Teilmodelle an, so verlieht das daraus entstehende Modell an Aussagekraft. Konnten zuvor Fehler direkt auf die betroffene Komponente zurückgeführt werden, so können nun nur noch Aussagen über die Anzahl der beschädigten Komponenten eines Typs getroffen werden.

Durch diese Einschränkung der Aussagekraft muss überlegt werden, an welche Stellen unseres Modells diese Form der Reduktion angewendet werden kann, ohne das Modell entscheidend zu verfälschen. Betrachtet man Abbildung 8.1 ist leicht zu erkennen, dass das Modell in verschiedene Abschnitte unterteilt werden kann. Zum einen den seriellen Teil bestehend aus dem Rechen, Sandfang und Vorklärbecken. Darauf folgt die Reinigungsstraße der parallel und in Reihe geschalteten Belebungsbecken, die einen eigenen, hier blau markierten Abschnitt bilden. Dieser Abschnitt wird gefolgt vom Nachklärbecken, das wieder zum seriellen Teil der Kläranlage gehört.

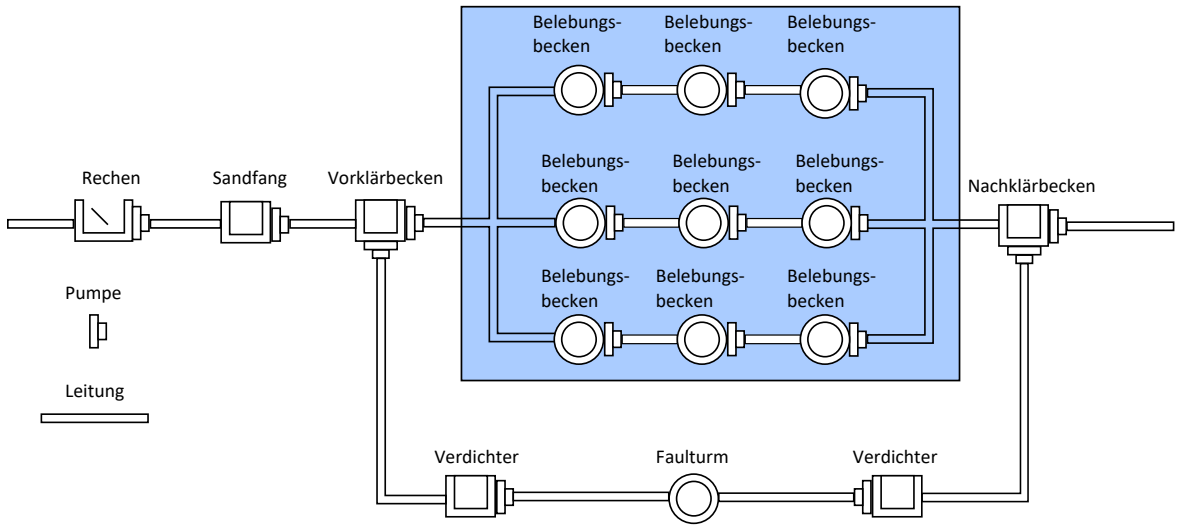

Abbildung 8.1: Schematische Darstellung der Kläranlage. Der parallel arbeitende Abschnitt der Anlage ist blau hinterlegt.

Parallel zu diesen Abschnitten verläuft zudem die Verarbeitung des Klärschlamms im primären- und sekundären Verdichter. Hierbei wird schnell offensichtlich das sich z.B. der Ausfall einer Pumpe zwischen dem Sandfang und Vorklärbecken im seriellen Abschnitt der Anlage stark von dem einer Pumpe unterscheidet, die sich im parallelen arbeitenden Teil der Anlage zwischen den Belebungsbecken befindet. Bei einem solchen Ausfall im parallelen Abschnitt wäre trotz des Ausfalls der Komponente ein Arbeiten der Anlage, wenn auch mit verringerter Leistung, möglich. Fällt eine der Komponenten im seriellen Teil der Anlage aus, so führt dies jedoch auf längere Sicht zu einem Ausfall der gesamten Anlage, bis die entsprechende Komponente repariert wurde.

Im Hinblick auf dieses Verhalten der Anlage anhand der sonst identischen Pumpen- und Leitungselemente sollte bei der Modellierung zwischen parallel und seriell geschalteten Elementen unterschieden werden.

Die Verwendung der Symmetrie Reduktion wird auch durch PRISM unterstützt [1]. Allerdings erlaubt der hier verwendete Befehl $-symm < before >< after >$ es lediglich an einer Stelle des Modells Komponenten mit gleichem Aufbau zusammenzufassen. Da in dem hier betrachteten Modell die Symmetrie Reduktion für verschiedene Gruppen von Komponenten, den Pumpen, Leitungen und Belebungsbecken mit einer zusätzlichen Unterscheidung zwischen parallelen und seriellen Komponenten, vorgenommen werden soll, kann die automatische Symmetrie Reduktion

von PRISM hier nicht verwendet werden. Aus diesem Grund wurde die Symmetrie Reduktion an dieser Stelle von Hand auf die einzelnen Komponentengruppen angewandt.

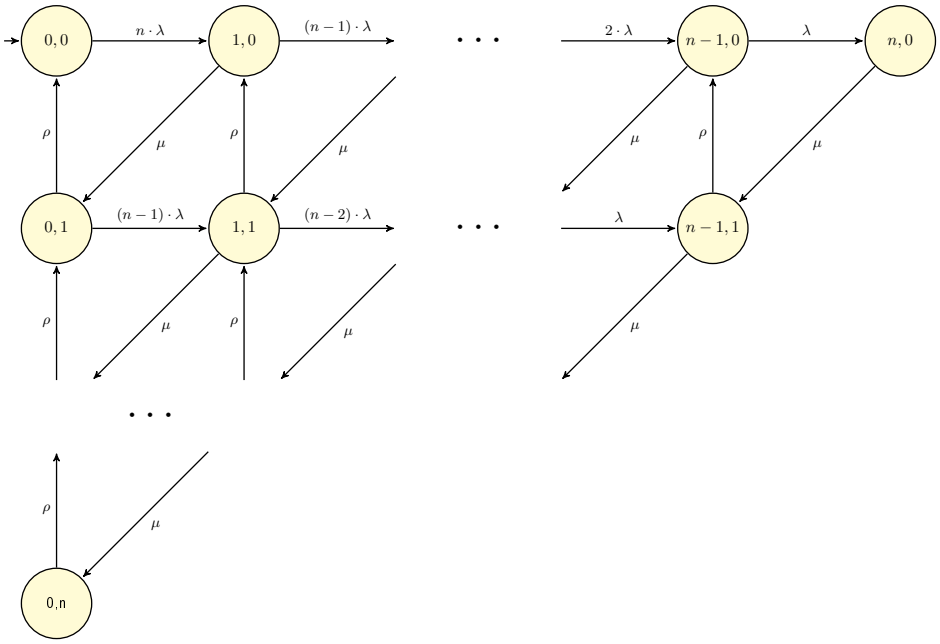

Abbildung 8.2: Aufbau der durch die Symmetrie Reduktion entstandenen CTMCs in allgemeiner Form mit der Beschädigungsrate λ, der Entdeckungsrate μ und der Reparaturrate ρ .

Hierbei unterscheiden wir zwischen 10 separaten Modellen für die jeweiligen Komponentengruppen. Ein Modell bilden die 8 seriell geschalteten Pumpen ab, ein weiteres die 8 seriellen Leitungen. Die Modellierung der Parallelen Komponenten ist hingegen komplexer. Es existieren insgesamt drei Reinigungsstraßen mit je drei Belebungsbecken drei Pumpen und zwei Leitungen. Da die drei Reinigungsstraßen parallel arbeiten, können immer jeweils drei identische Elemente zu einem Modell zusammengefasst werden. Es ergibt sich ein Modell für die ersten drei Belebungsbecken dann eines für die ersten drei Pumpen usw.. Insgesamt erhalten wir 8 Modelle, die jeweils drei der Komponenten abbilden. Schematisch ist der Aufbau dieser Modelle in Abbildung 8.2 zu sehen. Insgesamt erhalten wir durch die Symmetrie Reduktion nun ein neues wesentlich kleineres Gesamtmodell. Dieses besteht aus 16 separat modellierten CTMCs, die nach der parallelen Komposition ein Modell mit rund $2.46 \cdot 10^{15}$ Zuständen und $6.24 \cdot 10^{16}$ Transitionen ergeben. Dies stellt eine Verkleinerung des Modells um 8 Größenordnungen im Vergleich zum naiven Modell dar.

Wie allerdings bereits in Kapitel 7 beschrieben, ist dieses Modell immer noch entschieden zu groß, als dass PRISM in der Lage wäre, dieses Modell zu Checken. Somit muss das Modell noch auf andere Weise verkleinert werden.

8.3 Verbinden paralleler Elemente

Wie bereits bei der Modellierung der einzelnen CTMCs der Symmetrie Reduktion zu sehen war, stellt das Auftreten vieler paralleler Komponenten ein Problem bei der Symmetrie Reduktion dar. Viele der Komponenten durch ihr paralleles Auftreten nicht miteinander verbunden werden, da andernfalls die Logik des Modells versagen würde. Aus diesem Grund erschien es an dieser Stelle sinnvoll die parallelen Komponenten der Reinigungsstraßen zu abstrahieren und zusammenzufassen. Das Resultat dieser Überlegung ist in Abbildung 8.3 zu sehen.

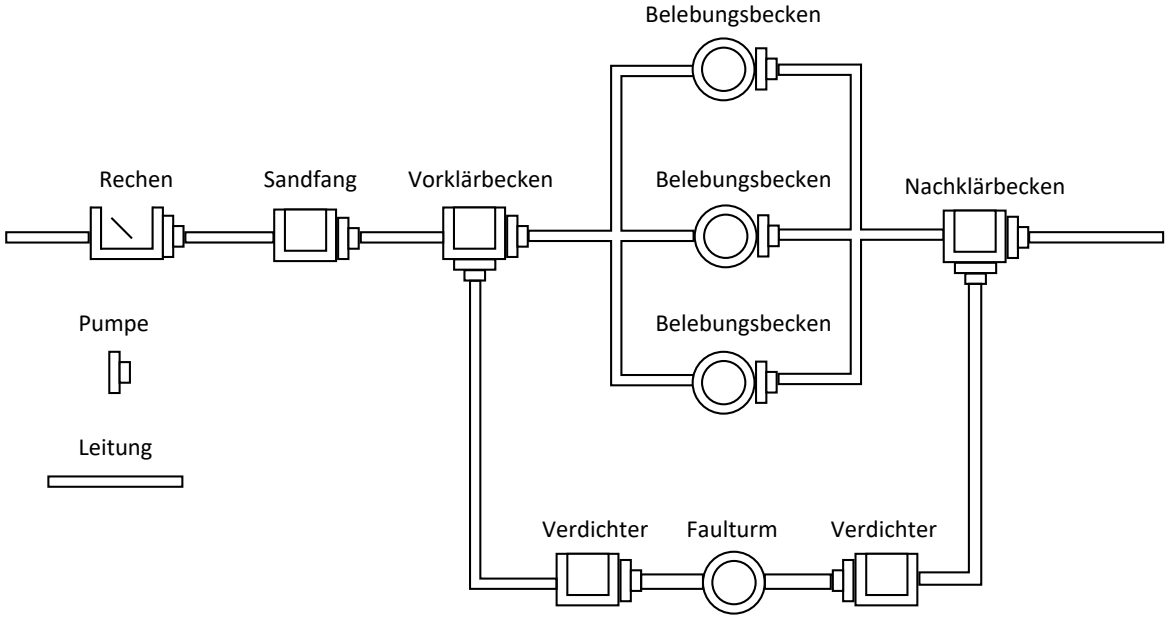

Abbildung 8.3: Schematische Darstellung der Kläranlage nach dem zusammenfügen der parallelen Komponenten.

Dieses besteht aus 16 separat modellierten CTMCs, die nach der parallelen Komposition ein Modell mit mit rund $2.46 \cdot 10^{15}$ Zuständen und $6.24 \cdot 10^{16}$ Transitionen ergeben.

Wie zu erkennen ist, wird nun jede der Reinigungsstraßen lediglich durch ein Belebungsbecken, eine Pumpe sowie eine Leitung repräsentiert. Um jedoch nicht den Aufbau der Kläranlage zu verfälschen, wurden an dieser Stelle die entsprechenden Raten der Komponenten angepasst. Abschnitt 4.2 beschreibt das allgemeine Verhalten von parallelen und seriellen Komponenten, in diesem Falls wurde eine starke Vereinfachung des Systems vorgenommen und lediglich eine*worst case* Betrachtung der einzelnen Komponenten vorgenommen. Hierbei wurden die entsprechenden Raten zur Beschädigung jeweils mit der Anzahl der zu repräsentierenden Komponenten multipliziert. Die Raten für die Detektion und Beschädigung blieben hingegen unverändert. Dies deckt somit den schlimmsten Fall ab der auftreten kann, da die Komponenten drei mal so schnell beschädigt werden, die Detektion und Reparatur jedoch gleich schnell abläuft.

Die so vorgenommene Änderung sorgt erneut für eine Verkleinerung des Modells um 5 Größenordnungen auf rund $2.46 \cdot 10^{10}$ Zustände und $4.03 \cdot 10^{11}$ Transitionen. Hiermit befinden wir uns schon sehr nahe an der von PRISM als noch berechenbar angesehenen Größenordnung von ca. $10^7 - 10^8$. Aus diesem Grund wurde versucht an dieser Stelle bereits Model Checking mit der MTBDD Engine Kapitel 6 von PRISM versucht. Diese ist laut den Entwicklern von PRISM auch in der Lage größere Modelle als die regulär verwendete Hybrid Engine zu bewältigen [14]. Allerdings war auch die MTBDD Engine nicht in der Lage, Model Checking auf dem Modell in dieser Größe durchzuführen. Somit musste das Modell erneut verkleinert werden.

8.4 Verbindungselemente

Wie in Abbildung 8.3 sehr gut zu erkennen ist, sind die einzelnen Becken der Kläranlage jeweils über eine Leitung sowie eine zugehörige Pumpe für den Transport des Wassers bzw. Klärschlamms verbunden. Im Folgenden wird versucht, jeweils eine Pumpe und eine Leitung abstrakt als ein Verbindungselement zwischen den einzelnen Klärbecken zu betrachten. Hierdurch ist es dann möglich die jeweiligen CTMCs der Leitungen und Pumpen ebenfalls durch eine einheitliche CTMC zu ersetzen die beide Elemente beschreibt.

Wie in Tabelle 7.1 zu sehen ist, besitzen Pumpen und Leitungen jeweils eine eigene voneinander unterschiedliche Beschädigungs- Detektions- und Reparaturrate. Diese müssen nun so miteinander in Verbindung gesetzt werden, dass das neue Modell möglichst wenig an seiner Aussagekraft gegenüber den beiden bisherigen Modellen verliert. Zunächst muss die Beschädigungsrate ermittelt werden. Eine Leitung wird nach Tabelle 4.2 durchschnittlich alle drei Monate beschädigt eine Pumpe hingegen alle sechs Monate. Es wird angenommen, dass diese beiden Ereignisse stochastisch unabhängig sind. Somit beeinflusst der Schaden an einer Pumpe weder die zugehörige Leitung noch ist dies in umgekehrter Form der Fall. Infolgedessen ist es uns möglich, die einzelnen Wahrscheinlichkeiten zu addieren. Ein Verbindungselement wird also durchschnittlich alle zwei Monate beschädigt. Hierbei kann es sich sowohl um einen Pumpen- als auch einen Leitungsschaden handeln. Nun muss ein solcher Schaden erkannt werden. Auch in diesem Fall wird davon ausgegangen, dass es sich bei der Erkennung der einzelnen fehlerhaften Elemente um stochastisch unabhängige Ereignisse handelt. Der Schaden einer Pumpe wird nach zwölf, der einer Leitung nach einer Stunde bemerkt. Somit wird ein Fehler durchschnittlich in $6\frac{1}{2}$ Stunden erkannt. Ebenso verhält es sich mit den Reparaturzeiten von zwei und einer Stunde. Also erhalten wir für die Verbindungselemente eine durchschnittliche Reparaturzeit von $1\frac{1}{2}$ Stunden.

Mit diesen Annahmen ist es uns nun möglich die jeweiligen CTMCs zur Beschreibung der seriellen Pumpen und Leitungen sowie der parallelen Pumpen und Leitungen jeweils zu einem Modell für serielle Verbindungselemente und paralleler Verbindungselemente zusammenzufügen. Nach dieser letzten Reduktion besitzt unser Gesamtmodell lediglich $5,46 \cdot 10^7$ Zustände und $6.67 \cdot 10^8$ Transitionen vergleiche Listing 8.1. Somit war es durch diese Abstraktion möglich, das Modell erneut um drei Größenordnungen zu verkleinern.

Nach all diesen Maßnahmen um die Größe des hier präsentierten Modells drastisch zu verkleinern haben wir ein Modell erschaffen, dass zugleich den Aufbau der Kläranlage, unter Berücksichtigung der FMEA Ergebnisse, möglichst realistisch abbildet und zugleich eine Größenordnung von Zuständen und Transitionen aufweist, die gering genug ist, sodass PRISM für das spätere Model Checking verwendet werden kann. Mit dem Prüfen und Verifizieren verschiedener Eigenschaften des Modells befassen wir uns im folgenden Kapitel zum Model Checking.

```
Building model...

Computing reachable states...

Reachability (BFS): 41 iterations in 0.01 seconds (average
    0.000317, setup 0.00)

Time for model construction: 0.071 seconds.

Type:        CTMC
States:      5467500 (1 initial)
Transitions: 66703500

Rate matrix: 1706 nodes (24 terminal), 66703500 minterms,
    vars: 27r/27c
```

Listing 8.1: Ausgabe des PRISM Logs bei der Erstellung des minimierten Modells.

9 Model Checking Ergebnisse

Dieses Kapitel befasst sich mit den Ergebnissen des Model Checkings, und wie es möglich ist, nach der Erstellung des Modells mögliche Schwächen im System aufzudecken. Hierzu zählt zunächst eine ausführliche Betrachtung des Steady State der Anlage, da dieser Aussagen über das Langzeitverhalten der Kläranlage liefert. Dies ist ein sehr interessanter Betrachtungswinkel, da Kläranlagen oft über Jahre und Jahrzehnte im Einsatz sind. Im Folgenden werden dann zwei Szenarien beschrieben, die mögliche Desaster die bei einer Kläranlage auftreten können beschreiben. Hierbei wird zunächst der Sachverhalt erläutert bevor er in Form einer CSL Formel ausgedrückt und geprüft wird.

9.1 Steady State Verhalten der Anlage

Um das Steady State Verhalten der Anlage auf sinnvolle Weise untersuchen zu können müssen die rund 5.4 Millionen Zustände des Modells zunächst mit Labeln versehen werden, die dieses logisch unterteilen. Eine mögliche Unterteilung besteht den gesamten Zustandsraum so zu unterteilen, wie es bereits in den einzelnen Modulen vorgegeben ist. Als in die Bereiche *"Stable"*, *"Damaged"* und *"Repairing"*. Diese spiegeln wie ihr Name bereits sagt einen Bereich wieder, in dem die Anlage ohne jegliche Fehler arbeitet, beschädigt ist, jedoch dieser Schaden noch nicht bemerkt wurde, sowie beschädigt ist und der besagte Schaden sich in Reparatur befindet. Diese Unterteilung des Zustandsraums ist wie folgt möglich. Der Zustand eines jeden Moduls, in dem es unbeschädigt ist, ist immer der Ausgangszustand, Zustand 0 des Moduls. Somit ist die Beschreibung eines stabilen Zustandes der CTMC sehr einfach und lässt sich wie folgt in PRISM ausdrücken:

```
label "Stable" = state_ce = 0 & state_ab=0 & state_pce = 0
    & state_bs = 0 & state_si = 0 & state_pst = 0 &
    state_sst = 0 & state_ps = 0 & state_ss = 0;
```

Listing 9.1: Beschreibung des "Stable" Labels mithilfe der PRISM Language.

Die Beschreibung der anderen Label erweist sich jedoch als etwas schwieriger, da bestimmt werden muss, wie genau die einzelnen Bereiche unterteilt werden. Z.B. stellt sich die Frage, zu welchem Label ein Zustand zählen soll, in dem sowohl 3 Komponenten beschädigt sind jedoch zugleich eine 4. repariert wird. Die Unterteilung wurde hierbei wie folgt vorgenommen. Solange es mindestens eine Komponente gibt, die beschädigt ist und zur gleichen Zeit keine Reparaturen stattfinden, so wurde dieser Zustand in den Bereich *"Damaged"* eingeordnet. Sobald ein oder mehrere Komponenten repariert werden unabhängig davon, wie viele andere unbemerkt beschädigt sind, wird dieser Zustand zu dem Label *"Repairing"* gezählt. Die sich aus dieser Zuweisung ergebenden Label in der PRISM Language sehen wie folgt aus:

```
label "Damaged" = ((state_ce > 0 & state_ce < 9) | (
    state_ab > 0 & state_ab < 4) | (state_pce > 0 &
    state_pce < 4) | state_bs = 1 |  state_si = 1 |
    state_pst = 1 | state_sst = 1 | state_sst = 3 | state_ps
     = 1 | state_ss = 1 ) & (state_ce < 9 & state_ab < 4 &
    state_pce < 4 & state_bs != 2 & state_si != 2 &
    state_pst != 2 & state_sst != 2 & state_sst != 4 &
    state_ps != 2 & state_ss != 2);

label "Repairing" = state_ce >= 9 | state_ab >= 4 |
    state_pce >= 4 | state_bs = 2 | state_si = 2| state_pst
    = 2 | state_sst = 2 | state_sst = 4 | state_ps = 2 |
    state_ss = 2;
```

Listing 9.2: Beschreibung der Label "Damaged" und "Repairing" mithilfe der PRISM Language.

Mit den so beschriebenen Labeln ist es nun möglich, die Zustände der CTMC in drei "Bereiche" zu unterteilen. Die nun zu beantwortende Frage ist, zu welchen Anteilen sich das System über lange zeit hin in den Bereichen befindet. Hierzu kann der Stady State Operator der CSL Logik verwendet werden. Bei der Prüfung der Formeln erlaubt

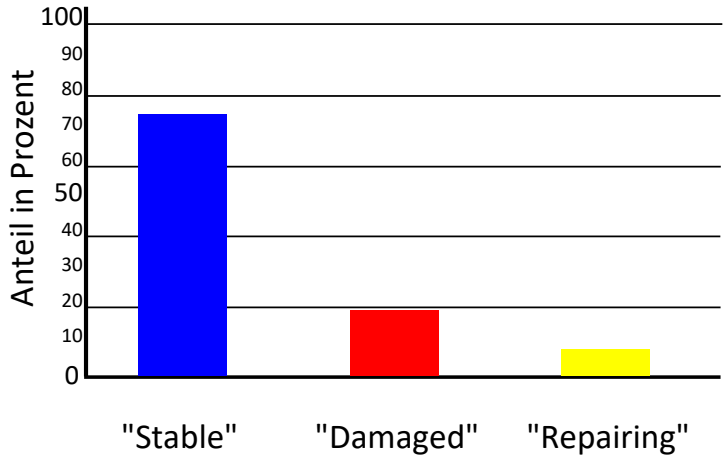

Abbildung 9.1: Prozentuale Aufteilung des Steady States der Label ” Stable ”, ” Damaged ” und ” Repairing ”.

PRISM auch die Angabe von Labeln als zu betrachtenden Bereich des Zustandsraums. Zudem möchten wir den genauen Anteil erfahren, zu dem sich die das System in den einzelnen Bereichen befindet. Somit ergeben sich für das Model Checking die Formeln S=? [” Stable ”], S=? [” Damaged ”] sowie S=? [” Repairing ”]. Die Ergebnisse dieser Formeln sind in Abbildung 9.1 zu sehen.

Wie zu erkennen ist, befindet sich die Kläranlage mit rund 74, 137% in einem stabilen, mit 18, 468% in einem beschädigten und mit 7, 393% in einem reparierenden Zustand. Hierbei ist zu bemerken, dass ein Schaden nicht direkt zum vollständigen Ausfall der Anlage führt. Jedoch befindet sich die Anlage mit über 18% in einem Bereich, in dem Schäden auftreten, jedoch nicht bemerkt wurden. DIes lässt darauf schließen, dass die Erkennungsrate von Fehlern bzw. die mit dem Auffinden von Fehlern verbundene Zeit in unserem Modell sehr hoch ist.

Die hier verwendeten Daten basieren wie bereits in Kapitel 7 beschrieben auf Schätzwerden. In einer realen Anwendung würden die nun erhaltenen Ergebnisse mit den Meinungen der an der FMEA Analyse beteiligten Experten verglichen und gegebenenfalls iterativ die Werte so lange angepasst, bis die Ergebnisse den Erwartungen entsprechen. Somit wird ein Modell erzeugt, dass möglichst genau die Wirklichkeit widerspiegelt und somit Aufschluss über mögliche Mängel im System geben kann. Da für diese Arbeit kein Expertenteam zur Verfügung steht, bleiben wir für weitere Betrachtungen bei den bisher angenommenen Werten.

Unter der Annahme, dass unsere Werte korrekt die Wirklichkeit widerspiegeln, würden die Ausfallwerte möglicherweise als zu hoch betrachtet werden. In diesem Fall muss nun die Ursache der hohen Ausfallrate und Reparaturzeiten gefunden werden. Hierfür müssen die Werte auf die einzelnen Komponenten aufgeteilt werden. Dies lässt sich leicht durch die Erstellung neuer Label auf Basis der bereits bestehenden erreichen. Mit den folgenden Labeln lässt sich z. B. die Beschädigungs- und Reparaturzeiten des Rechen bestimmen:

```
label "Damaged_bs" = state_ce = 0 & state_ab = 0 &
    state_pce = 0 & state_bs = 1 & state_si = 0 & state_pst
    = 0 & state_sst = 0 & state_ps = 0 & state_ss = 0;

label "Repairing_bs" = state_bs = 2;
```

Listing 9.3: Label für die Beschädigungs- und Reparaturzeiten des Rechen.

Hierbei ist zu beachten, dass nur Beschädigungen einzelner Komponenten betrachtet werden, jedoch nicht die Kombination verschiedener. Die Ergebnisse dieser Betrachtung sind in Abbildung 9.2 und Abbildung 9.3 zu sehen. Bei der Beschädigung von Komponenten fallen besonders die hohen Werte für die parallelen Verbindungselemente (37.321%), die Verbindungselemente (16.223%) sowie der Belebungsbecken (11.061%) auf. Hierbei ist zu beachten, das diese Komponentengruppen jeweils aus vielen Komponenten zusammengesetzt sind. So sind unter den parallelen Verbindungselementen sowohl 9 separate Pumpen als auch Verbindungen die dieser Abschnitt repräsentiert. Jedem dieser Elemente fällt somit durchschnittlich nur rund 2.073% der Zeit zu. Auch die Gruppe der Verbindungselemente umfasst wie bereits zuvor beschrieben 16 Pumpen und Leitungen. Die Gruppe der Belebungsbecken besteht hingegen aus 9 separaten Becken. Auf diese Weise lassen sich die hohen Werte für diese drei Gruppen erklären. Die verhältnismäßig hohen Anteile des Rechen(8.345%) sowie der beiden Verdichter(je 8.941% lassen sich durch die geringe Erkennungsrate von Schäden bei diesen Becken erklären. Sie werden nicht übermäßig oft beschädigt, jedoch verbringen sie jeweils eine längere Zeit im beschädigten Zustand, bis der Schaden entdeckt und repariert werden kann.

Bei den Reparaturzeiten Fallen besonders die Belebungsbecken mit anteilig 53.787% der Reparaturzeit auf. Für die Becken wie auch die beiden Typen der Verbindungselemente gilt wieder, dass sie jeweils mehrere Komponenten gleichzeitig modellieren. Zudem ist die Reparaturzeit der Belebungsbecken mit durchschnittlich 10 Stunden

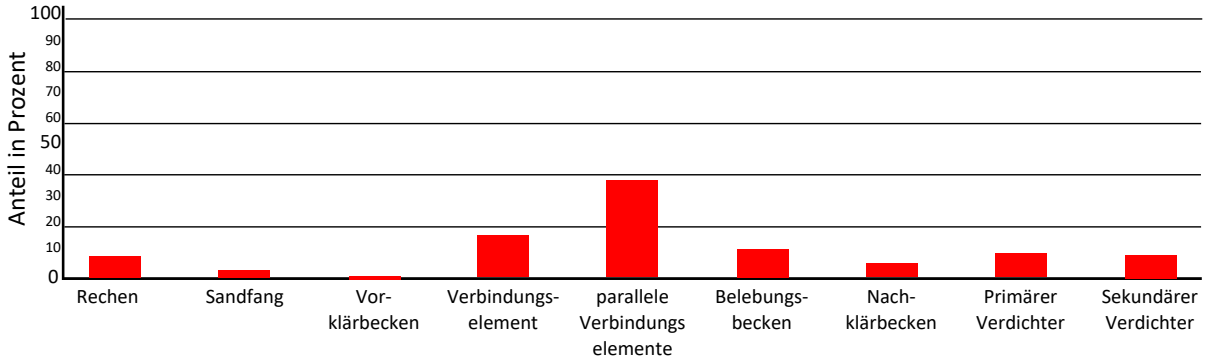

Abbildung 9.2: Prozentuale Verteilung, welche Elemente am häufigsten beschädigt werden.

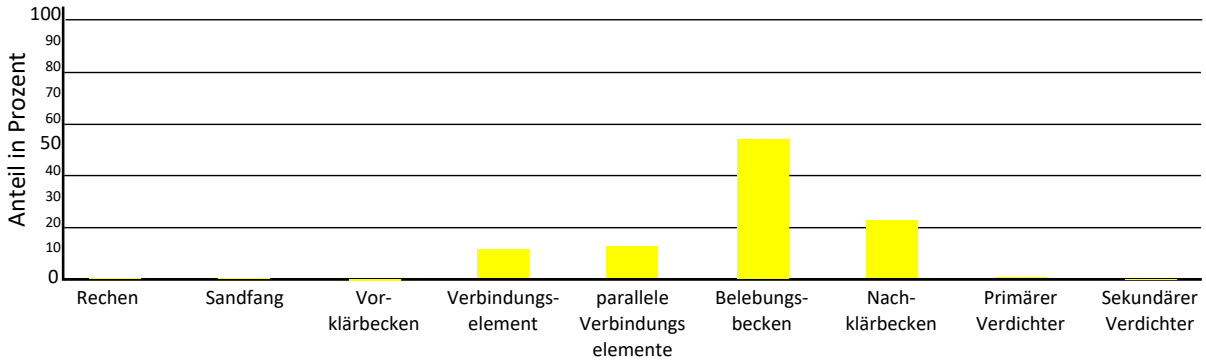

Abbildung 9.3: Prozentuale Verteilung, welche Elemente am häufigsten repariert werden.

in unserer vorherigen Analyse der höchste gewählte Wert für eine Reparatur, da z.B. der Austausch von Bakterienkulturen oder die Reparatur der Belüftungsanlagen oft sehr zeitspielig ist. Ebenfalls mit einer durchschnittlichen Reparaturzeit von 10 Stunden versehen ist das Nachklärbecken, welches mit anteilig 22.305% ebenfalls in der Statistik hervorsticht. Hierbei ist zu bedenken, dass im Nachklärbecken sowohl eine normale Beschädigung, als auch das mögliche Auftreten von Schwimmschlamm modelliert wurde. Da somit zwei Fehlerarten mit jeweils sehr hohen Reparaturzeiten entstehen, lässt sich der entsprechend hohe Wert in dieser Betrachtung ebenfalls erklären.

Insgesamt spiegeln sowohl der Werte der Beschädigungs- als auch der Reparaturzeiten die erwarteten Werte wieder. In diesem Teil der Analyse wäre es nun möglich, falls Ausreißer im System auftreten sowohl die Modellierung des Systems selbst als auch die fraglichen Komponenten einer genaueren Untersuchung zu unterziehen.

Sollte das System den Erwartungen entsprechen so müssen unter Umständen verwendete Komponenten durch robustere Varianten ersetzt werden oder zusätzliche Erkennungs- und Meldeverfahren für Fehler und Beschädigungen eingerichtet werden, um die gewünschte Sicherheit der Anlage zu gewährleisten. Im Folgenden werden zwei Szenarios vorgestellt, welche mögliche Desaster beschreiben, die in einer Kläranlage auftreten können, sowie diese mithilfe der CSL Logik modelliert und geprüft werden können.

9.2 Szenario 1

Das erste Szenario, dass hier betrachtet wird, ist die Bildung von Schwimmschlamm im Nachklärbecken, welches dann auf die Belebungsbecken übergreift. Zur Modellierung eines solchen Desasters ist es möglich, den Ausgangszustand des Modells zu verändern. Somit startet das System bereits in dem Zustand des Desasters und es kann geprüft werden, wie lange es dauert, bis sich die Anlage aus diesem Zustand erholt. Im Fall der Schwimmschlammbildung setzen wir hierbei das Nachklärbecken auf den bereits dafür modellierten Zustand sowie alle Belebungsbecken in einen ”beschädigt” Zustand. Hierbei ist zu beachten, dass die Kläranlage bereits dann wieder einsatzbereit ist, wenn eine der Klärstraße bestehend aus 3 Belebungsbecken sowie das Nachklärbecken wieder einsatzbereit sind. Somit können bei der *Recovery* zwei verschiedene Fälle betrachtet werden. Zum einen die Recovery Rate, die benötigt wird, bis das System wieder seinen Betrieb aufnehmen kann sowie die, bis das System wieder vollkommen funktionsfähig ist. Zudem lassen sich vorab Zeiten definieren, mit denen angegeben werden kann, welche Reparaturzeiten als akzeptabel zu betrachten sind, da das System für diese Zeitdauer ausfallen kann, ohne das der Endnutzer den Fehler bemerkt, weil z.B. ein anderes Klärwerk vorübergehend die Last der ausgefallenen Kläranlage mit trägt.

Mit der CSL Logik lässt sich dieser Sachverhalt als Time Bounded Until Formel ausdrücken. Der Time Bound gibt an, wie viel Zeit verstreicht. Der so bestimmte prozentuale Wert gibt dann an, mit welcher Wahrscheinlichkeit das System nach dieser Zeit wieder in einem bestimmten Zustand ist. Die so entwickelten Formeln sehen wie folgt aus:

```
P=? [true U<=x "Operational"]
```

```
P=? [true U<=x "Stable"]
```

Listing 9.4: CSL Formeln für die Recovery der Kläranlage nach der Schwimmschlammbildung.

Hierbei wurde das Label `Operational` verwendet, dieses beschreibt, dass mindestens eine der Klärstraßen sowie der Nachklärbecken einsatzbereit sind und der Wert x die jeweils gewählte Zeitspanne, die vergangen ist. Abbildung 9.4 zeigt die Wahrscheinlichkeit, dass die Anlage nach einer gegebenen Zeit wieder eines der beiden Servicelevel `Operational` oder `Stable` erreicht. Hierbei wurde der Wert der Until Formel jeweils zu jeder Stunde geprüft. Der Verlauf zwischen den Messwerten wurde dann approximiert. Wie zu erkennen ist, handelt es sich bei diesem Szenario um ein sehr gravierendes Desaster, welches mehrere Tage benötigt bis sich die Anlage von dem Vorfall wieder erholt. Nach etwa 48 Stunden sollte die Anlage wieder einsatzbereit sein. Bis jedoch die volle Leistung der Anlage wieder erreicht wird vergehen jedoch wahrscheinlich mehr als 80 Stunden. Auch die für die Berechnung der Until Formeln benötigte Zeit ist sehr unterschiedlich und steigen stetig an, da für längere Zeitspannen mehr Iterationen der Uniformisation Berechnung benötigt werden. Die Berechnungen wurden auf einem Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz mit 8 GB Arbeitsspeicher durchgeführt. Tabelle 9.1 zeigt die benötigte Zeit zur Berechnung einiger der Until Formeln sowie die Anzahl der benötigten Iterationen der Uniformisation. Da die Berechnungszeiten und Anzahl Iterationen für die beiden Label identisch sind wurden hier nur die Werte einer der Berechnungen aufgelistet

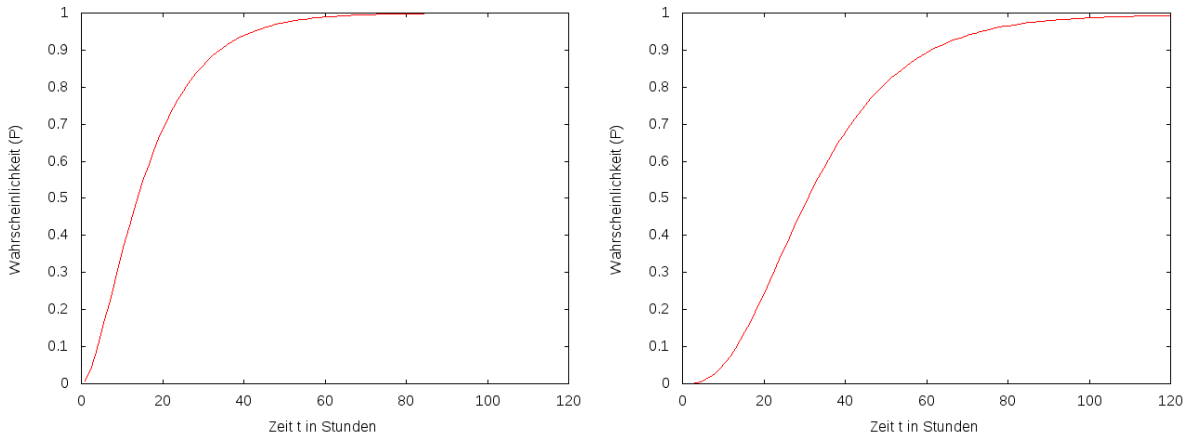

Abbildung 9.4: Wahrscheinlichkeit (P), dass das Servicelevel Operational (links) oder Stable(rechts) nach t Stunden wiederhergestellt werden kann.

Formel	Berechnungsdauer	Anzahl
P=? [true U<=1/24 Operational]	18.29 sek	19
P=? [true U<=12/24 Operational]	45.15 sek	96
P=? [true U<=24/24 Operational]	68.86 sek	165
P=? [true U<=48/24 Operational]	112.84 sek	294
P=? [true U<=72/24 Operational]	157.04 sek	417
P=? [true U<=96/24 Operational]	218.84 sek	601
P=? [true U<=120/24 Operational]	255.54 sek	670

Tabelle 9.1: Auflistung verschiedener Until Formeln mit der jeweiligen Berechungsdauer und Anzahl Iterationen pro Formel.

Ein solches Szenario kann nun Aufschluss darüber geben, dass zusätzliche Sicherheitsmaßnahmen getroffen werden müssen, um ein solches Szenario präventiv zu verhindern. Dies ist auch der Fall, da in Kläranlagen mehrmals täglich der Bakteriengehalt im Wasser der Belebungs- und Nachklärbecken gemessen wird, um die Bildung von Schwimmschlamm zu verhindern.

9.3 Szenario 2

Ein weiteres Szenario, das hier betrachtet wird, ist die Beschädigung der Kläranlage als Folge von starken Regenfällen. Durch anhaltende starke Regenfälle werden große Teile der Kanalisation mit wesentlich mehr Wasser durchflossen wird, als es an regulären Tagen der Fall ist. Dies führt nicht nur dazu, dass die Kläranlagen wesentlich mehr Wasser in kurzer Zeit aufnehmen müssen, sondern kann auch weitere Folgen haben. Durch die enormen Wassermassen besteht die Möglichkeit, dass große Steine und Äste die zuvor in der Kanalisation lagen und nicht durch das Abwasser beeinflusst wurden nun mit großer Wucht zur Kläranlage gespült werden. Dies kann zu Schäden des Rechens und Sandfangs führen, da die Steine durch ihre Wucht den Rechen direkt beschädigen und durch diese Beschädigung grober Schmutz wie Steine und Äste, die zuvor aussortiert wurden, in den Sandfang gelangen und auch diesen beschädigen.

Die Modellierung dieses Szenarios erfolgt sehr ähnlich zu der des Ersten. Zunächst wird der Initialzustand des Systems auf die bestehende Beschädigung eingestellt. Im Folgenden kann dann erneut die Wahrscheinlichkeit geprüft werden, dass sich das System in einem gewissen Zeitraum wieder erholt. Die hierbei verwendete CSL Formel

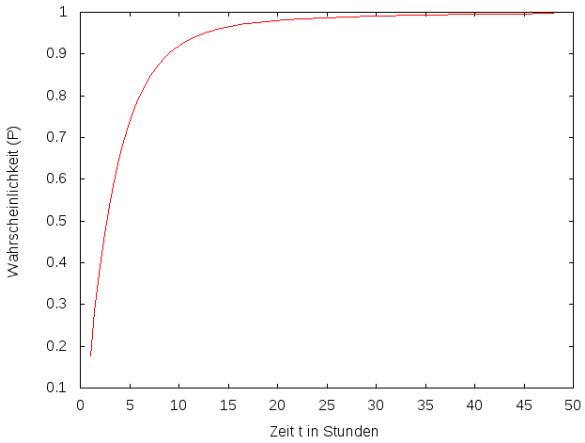

Abbildung 9.5: Wahrscheinlichkeit (P), dass das Servicelevel Stable nach t Stunden wiederhergestellt werden kann.

Formel	Berechnungsdauer	Anzahl
P=? [true U<=1/24 Stable]	17.46 sek	19
P=? [true U<=6/24 Stable]	31.54 sek	58
P=? [true U<=12/24 Stable]	46.99 sek	96
P=? [true U<=24/24 Stable]	75.19 sek	165
P=? [true U<=32/24 Stable]	93.08 sek	209
P=? [true U<=48/24 Stable]	127.64 sek	294
P=? [true U<=60/24 Stable]	152.959 sek	356

Tabelle 9.2: Auflistung verschiedener Until Formeln mit der jeweiligen Berechungsdauer und Anzahl Iterationen pro Formel.

ist analog zu der in Listing 9.4 beschriebenen Formel, allerdings gibt es in diesem Fall keine verschiedenen Level der Recovery, da es jeweils nur einen Rechen und einen Sandfang in der Anlage gibt und beide einsatzbereit sein müssen, damit das Gesamtsystem funktionieren kann.

Abbildung 9.5 zeigt die Zeit, die die Anlage benötigt um wieder in einen funktionsfähigen Zustand zu gelangen. Wie zu erkennen ist, ist eine Erholung von diesem Desaster im Gegensatz zu dem vorherig beschriebenen schon in einigen Stunden möglich. So ist die Anlage bereits nach 12 bis 16 Stunden fast sicher wieder einsatzbereit, welches im Verhältnis zum vorherigen Szenario wesentlich geringer ist. Jedoch ist hierbei zu bemerken, dass die Beschädigung der Anlage infolge starker Regenfälle ausgelöst wurde. Somit besteht die Möglichkeit, dass für die Zeit der Beschädigung keine anderen Anlagen die Last übernehmen können, da diese ebenfalls stark ausgelastet sind. Auch hier wäre eine Expertenmeinung gefragt, wie mit dem Problem umgegangen werden

soll. Die benötigte Berechnungszeit der Formeln, sowie die Anzahl der Iterationen ist in Tabelle 9.2 zu sehen. Wie zu erkennen ist, ist die Anzahl der benötigten Iterationen identisch zu denen die wir in Szenario 1 erhalten haben, da die Anzahl lediglich vom gewählten Timebound abhängt. Jedoch ist zu erkennen, das die Berechnungszeiten der einzelnen Formeln, wenn auch nur um einige Sekunden, merklich höher ist als die des vorherigen Szenarios.

Die beiden hier beschriebenen Szenarien können somit dazu beitragen mögliche Fehler in der Anlage aufzudecken oder bei der Erstellung von Notfallplänen zu helfen. Des Weiteren können sie Ausschluss über mangelnde Prüf- und Kontrollverfahren liefern, die benötigt werden, um das Auftreten solcher Desaster zu verhindern.

10 Fazit

In dieser Arbeit wurde ein Verfahren vorgestellt, mit dem aus den Ergebnissen einer FMEA Analyse ein CTMC Modell erzeugt werden kann, welches als Basis für Zuverlässigkeitsanalysen mithilfe der CSL Logik dient. Dies ermöglicht es genauere Aussagen über das Verhalten von Kläranlagen zu treffen, deren Zuverlässigkeit für eine stabile Grundversorgung mit Trinkwasser unerlässlich sind, als es mit einer reinen FMEA Analyse möglich ist. Zudem wurden verschiedene Verfahren zur Minimierung von CTMC Modellen untersucht und auf das Modell der Kläranlage in Enschede, welches als Fallstudie dient, angewendet. Des Weiteren wurde die Zuverlässigkeit des Modells anhand seines Steady States Verhaltens sowie zweier möglicher Szenarien untersucht, welche mögliche Notfallsituationen die in einem Klärwerk auftreten können modellieren. Hierbei wurde besonders auf das Konzept bei der Erstellung des Verfahrens eingegangen um eine Überführung auf andere Anlagen und Produkte zu ermöglichen, zu denen bereits FMEA Analysen vorliegen.

Weiterführende Arbeit in diesem Bereich könnte in viel Richtungen führen. Besonders interessant wäre hierbei ein Abgleich der verwendeten FMEA Daten und getroffenen Annahmen mit dem Wissen von Experten um die Genauigkeit der Untersuchung zu verifizieren. Weiterhin könnte versucht werden, die hier beschriebene Methodik auf andere Anwendungen und Produkte anzuwenden für die bereits FMEA Analysen existieren oder die sich noch in der Planungsphase befinden. Ein weiterer Punkt, indem weitere Arbeit möglich ist, ist die Minimierung der CTMC Modelle, welche zum einen automatisiert und zum anderen verbessert werden könnte, um die Modellierung noch größerer oder detaillierterer Anwendungen zu ermöglichen.

Abbildungsverzeichnis

Literaturverzeichnis

[1] *PRISM Homepage*. http://www.prismmodelchecker.org/.

[2] *PRISM Synchronisation*. http://www.prismmodelchecker.org/manual/ThePRISMLanguage/ProcessAlgebraOperators.

[3] *Ruhrverband Wissen, Werte, Wasser*. http://www.ruhrverband.de/de/abwasser/klaeranlagen/.

[4] BAIER, CHRISTEL, BOUDEWIJN HAVERKORT, HOLGER HERMANNS und J-P KATOEN: *Model-checking algorithms for continuous-time Markov chains*. IEEE Transactions on Software Engineering, 29(6):524–541, 2003.

[5] DOMINGUEZ-CHICAS, ANGELINA und MARK D. SCRIMSHAW: *Hazard and risk assessment for indirect potable re-use schemes: an approach for use in developing Water Safety Plans*. 2011.

[6] GHASEMIEH, HAMED, BOUDEWIJN R. HAVERKORT und ANNE REMKE: *Benchmark of a pipelining system: A Water Sewage Facility*. University of Twente, The Netherlands, University of Münster, Germany.

[7] HSIEH, MEI HUAN: *Application of Failure Mode and Effect Analysis based on Fuzzy Theory-The Case of Sewage Treatment Plant*. Diplomarbeit, 2005.

[8] KATOEN, J.-P., T. KEMNA, I.S. ZAPREEV und D.N. JANSEN: *Bisimulation minimisation mostly speeds up probabilistic model checking*. In: GRUMBERG, O. und M. HUTH (Herausgeber): *Tools and algorithms for the construction and analysis of systems*, Band 4424 der Reihe *Lecture notes in computer science*, Seiten 87–101, Berlin, 2007. Springer.

[9] KWIATKOWSKA, M., G. NORMAN und D. PARKER: *Probabilistic Verification of Herman's Self-Stabilisation Algorithm*. Formal Aspects of Computing, 24(4):661–670, 2012.

[10] KWIATKOWSKA, M., G. NORMAN und J. SPROSTON: *Probabilistic Model Checking of Deadline Properties in the IEEE 1394 FireWire Root Contention Protocol*. Formal Aspects of Computing, 14(3):295–318, 2003.

[11] KWIATKOWSKA, MARTA, NORMAN GETHIN und DAVID PARKER: *Symmetry Reduction for Probabilistic Model Checking*. In: *Computer Aided Verification*, 4144, Seiten pp 234–248, Berlin, Heidelberg, 2006. Springer-Verlag.

[12] KWIATKOWSKA, MARTA, GETHIN NORMAN und DAVID PARKER: *Stochastic Model Checking*. In: *Proceedings of the 7th International Conference on Formal Methods for Performance Evaluation*, SFM'07, Seiten 220–270, Berlin, Heidelberg, 2007. Springer-Verlag.

[13] LECHEVALLIER, MARK W und KWOK-KEUNG AU: *Water Treatment and Pathogen Control: Process Efficiency in Achieving Safe Drinking Water*. 20004.

[14] M. KWIATKOWSKA, G. NORMAN und D. PARKER: *PRISM 4.0: Verification of Probabilistic Real-time Systems*. In: GOPALAKRISHNAN, G. und S. QADEER (Herausgeber): *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, Band 6806 der Reihe *LNCS*, Seiten 585–591. Springer, 2011.

[15] PAUL, PRATHEEBA: *Optimization for maintenance of water distribution systems using stochastic search algorithms*. Faculty of civil engineering anna university, 2011.

[16] ROOLVINK, STEPHAN, ANNE REMKE und MARIËLLE STOELINGA: *Dependability and Survivability Evaluation of a Water Distribution Process with Arcade*. In: *PMCCS 2009*, Seiten 4–7, Budapest, September 2009. Budapest University of Technology and Economics.

[17] ROY BILLINTON, RONALD N. ALLAN: *Reliability Evaluation of Engineering Systems*. Springer US, 1992.

[18] STEWART., W. J.: *Introduction to the Numerical Solution of Markov Chains*. 1994.

[19] TABESH, M., J. SOLTANI, R. FARMANI und D. SAVIC: *Assessing pipe failure rate and mechanical reliability of water distribution networks using data-driven modeling*. Journal of Hydroinformatics, 11(1):1–17, 2009.

[20] ZAPREEV, I. S.: *Model Checking Markov Chains: Techniques and Tools*. Doktorarbeit, University of Twente, Enschede, The Netherlands, 2008.

Eidesstattliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit über *Analyse sicherheitskritischer Aspekte einer Kläranlage mit Methoden des quantitativen Model Checkings* selbstständig verfasst worden ist, dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt worden sind und dass die Stellen der Arbeit, die anderen Werken – auch elektronischen Medien – dem Wortlaut oder Sinn nach entnommen wurden, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht worden sind.

David Könning, Münster, 30. November 2016

Ich erkläre mich mit einem Abgleich der Arbeit mit anderen Texten zwecks Auffindung von Übereinstimmungen sowie mit einer zu diesem Zweck vorzunehmenden Speicherung der Arbeit in eine Datenbank einverstanden.

David Könning, Münster, 30. November 2016